

TRAITE DE COOPERATION EN MATIERE DE BREVETS

PCT

NOTIFICATION D'ELECTION

(règle 61.2 du PCT)

Expéditeur: le BUREAU INTERNATIONAL

Destinataire:

Commissioner
 US Department of Commerce
 United States Patent and Trademark
 Office, PCT
 2011 South Clark Place Room
 CP2/5C24
 Arlington, VA 22202
 ETATS-UNIS D'AMERIQUE
 en sa qualité d'office élu

Date d'expédition (jour/mois/année) 06 juin 2001 (06.06.01)	
Demande internationale no PCT/FR00/02349	Référence du dossier du déposant ou du mandataire BCT000077
Date du dépôt international (jour/mois/année) 21 août 2000 (21.08.00)	Date de priorité (jour/mois/année) 23 août 1999 (23.08.99)
Déposant LEROY, Xavier	

1. L'office désigné est avisé de son élection qui a été faite:

☒ dans la demande d'examen préliminaire international présentée à l'administration chargée de l'examen préliminaire international le:

20 mars 2001 (20.03.01)

☐ dans une déclaration visant une élection ultérieure déposée auprès du Bureau international le:

2. L'élection ☒ a été faite

☐ n'a pas été faite

avant l'expiration d'un délai de 19 mois à compter de la date de priorité ou, lorsque la règle 32 s'applique, dans le délai visé à la règle 32.2b).

Bureau international de l'OMPI
 34, chemin des Colombettes
 1211 Genève 20, Suisse

no de télécopieur: (41-22) 740.14.35

Fonctionnaire autorisé

Maria Kirchner

no de téléphone: (41-22) 338.83.38

THIS PAGE BLANK (11/11)

TRAITE D'COOPERATION EN MATIERE DE BREVETS

PCT

Expéditeur: le BUREAU INTERNATIONAL

NOTIFICATION DE L'ENREGISTREMENT
D'UN CHANGEMENT(règle 92bis.1 et
instruction administrative 422 du PCT)

Destinataire:

FRECHEDE, Michel
Cabinet Plasseraud
84, rue d'Amsterdam
F-75440 Paris Cedex 09
FRANCE

Date d'expédition (jour/mois/année) 30 avril 2001 (30.04.01)	
Référence du dossier du déposant ou du mandataire BCT000077	NOTIFICATION IMPORTANTE
Demande internationale no PCT/FR00/02349	Date du dépôt international (jour/mois/année) 21 août 2000 (21.08.00)

1. Les renseignements suivants étaient enregistrés en ce qui concerne:

☒ le déposant

 ☐ l'inventeur

 ☐ le mandataire

 ☐ le représentant commun

Nom et adresse TRUSTED LOGIC 23, avenue de Fulpmes F-78450 Villepreux FRANCE	Nationalité (nom de l'Etat) FR	Domicile (nom de l'Etat) FR
	no de téléphone	
	no de télécopieur	
	no de téléimprimeur	

2. Le Bureau international notifie au déposant que le changement indiqué ci-après a été enregistré en ce qui concerne:

☐ la personne

 ☐ le nom

 ☒ l'adresse

 ☐ la nationalité

 ☐ le domicile

Nom et adresse TRUSTED LOGIC 5, rue du Bailliage F-78000 Versailles FRANCE	Nationalité (nom de l'Etat) FR	Domicile (nom de l'Etat) FR
	no de téléphone	
	no de télécopieur	
	no de téléimprimeur	

3. Observations complémentaires, le cas échéant:

4. Une copie de cette notification a été envoyée:

<input checked="" type="checkbox"/> à l'office récepteur	<input checked="" type="checkbox"/> aux offices désignés concernés
<input type="checkbox"/> à l'administration chargée de la recherche internationale	<input type="checkbox"/> aux offices élus concernés
<input type="checkbox"/> à l'administration chargée de l'examen préliminaire international	<input type="checkbox"/> autre destinataire:

Bureau international de l'OMPI 34, chemin des Colombettes 1211 Genève 20, Suisse no de télécopieur (41-22) 740.14.35	Fonctionnaire autorisé: Dorothee Mülhausen no de téléphone (41-22) 338.83.38
---	--

THIS PAGE BLANK (USPTO)

TRAITE DE COOPERATION EN MATIERE DE BREVETS

PCT

NOTIFICATION DE L'ENREGISTREMENT
D'UN CHANGEMENT(règle 92bis.1 et
instruction administrative 422 du PCT)

Expéditeur: le BUREAU INTERNATIONAL

Destinataire:

FRECHEDE, Michel
Cabinet Plasseraud
84, rue d'Amsterdam
F-75440 Paris Cedex 09
FRANCERECU LE
07. MAI 2001
Cbt Plasseraud

Date d'expédition (jour/mois/année) 30 avril 2001 (30.04.01)	NOTIFICATION IMPORTANTE
Référence du dossier du déposant ou du mandataire BCT000077	
Demande internationale no PCT/FR00/02349	Date du dépôt international (jour/mois/année) 21 août 2000 (21.08.00)

1. Les renseignements suivants étaient enregistrés en ce qui concerne: --

☒ le déposant ☐ l'inventeur ☐ le mandataire ☐ le représentant commun

Nom et adresse

TRUSTED LOGIC
23, avenue de Fulpmes
F-78450 Villepreux
FRANCE

Nationalité (nom de l'Etat)

FR

Domicile (nom de l'Etat)

FR

no de téléphone

no de télécopieur

no de téléimprimeur

2. Le Bureau international notifie au déposant que le changement indiqué ci-après a été enregistré en ce qui concerne:

☐ la personne ☐ le nom ☒ l'adresse ☐ la nationalité ☐ le domicile

Nom et adresse

TRUSTED LOGIC
5, rue du Bailliage
F-78000 Versailles
FRANCE

Nationalité (nom de l'Etat)

FR

Domicile (nom de l'Etat)

FR

no de téléphone

no de télécopieur

no de téléimprimeur

3. Observations complémentaires, le cas échéant:

4. Une copie de cette notification a été envoyée:

☒ à l'office récepteur ☒ aux offices désignés concernés
☐ à l'administration chargée de la recherche internationale ☐ aux offices élus concernés
☐ à l'administration chargée de l'examen préliminaire international ☐ autre destinataire:
Bureau international de l'OMPI
34, chemin des Colombettes
1211 Genève 20, Suisse

no de télécopieur (41-22) 740.14.35

Fonctionnaire autorisé:

Dorothee Mülhausen

no de téléphone (41-22) 338.83.38

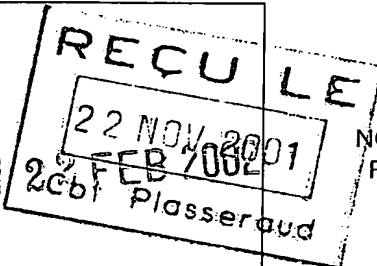
THIS PAGE BLANK (USPTO)

TRAITE DE COOPERATION EN MATIERE DE BREVETS

Expéditeur: L'ADMINISTRATION CHARGÉE DE
L'EXAMEN PRELIMINAIRE INTERNATIONAL

Destinataire:

FREDECHE, Michel et al.
CABINET PLASSERAUD
84, rue d'Amsterdam
75440 Paris cedex 09
FRANCE



PCT

NOTIFICATION DE TRANSMISSION DU
RAPPORT D'EXAMEN PRELIMINAIRE
INTERNATIONAL
(règle 71.1 du PCT)

Date d'expédition
(jour/mois/année) 20.11.2001

Référence du dossier du déposant ou du mandataire
BCT000077

NOTIFICATION IMPORTANTE

Demande internationale No.
PCT/FR00/02349

Date du dépôt international (jour/mois/année)
21/08/2000

Date de priorité (jour/mois/année)
23/08/1999

Déposant
TRUSTED LOGIC et al.

1. Il est notifié au déposant que l'administration chargée de l'examen préliminaire international a établi le rapport d'examen préliminaire international pour la demande internationale et le lui transmet ci-joint, accompagné, le cas échéant, de ces annexes.
2. Une copie du présent rapport et, le cas échéant, de ses annexes est transmise au Bureau international pour communication à tous les offices élus.
3. Si tel ou tel office élu l'exige, le Bureau international établira une traduction en langue anglaise du rapport (à l'exclusion des annexes de celui-ci) et la transmettra aux offices intéressés.

4. RAPPEL

Pour aborder la phase nationale auprès de chaque office élu, le déposant doit accomplir certains actes (dépôt de traduction et paiement des taxes nationales) dans le délai de 30 mois à compter de la date de priorité (ou plus tard pour ce qui concerne certains offices) (article 39.1) (voir aussi le rappel envoyé par le Bureau international dans le formulaire PCT/IB/301).

Lorsqu'une traduction de la demande internationale doit être remise à un office élu, elle doit comporter la traduction de toute annexe du rapport d'examen préliminaire international. Il appartient au déposant d'établir la traduction en question et de la remettre directement à chaque office élu intéressé.

Pour plus de précisions en ce qui concerne les délais applicables et les exigences des offices élus, voir le Volume II du Guide du déposant du PCT.

Nom et adresse postale de l'administration chargée de l'examen préliminaire international



Office européen des brevets
D-80298 Munich
Tél. +49 89 2399 - 0 Tx: 523656 epmu d
Fax: +49 89 2399 - 4465

Fonctionnaire autorisé

Schall, H

Tél. +49 89 2399-2647



THIS PAGE BLANK (USPTO)

TRAITE DE COOPERATION EN MATIERE DE BREVETS

PCT

RAPPORT D'EXAMEN PRELIMINAIRE INTERNATIONAL

(article 36 et règle 70 du PCT)

Référence du dossier du déposant ou du mandataire BCT000077	POUR SUITE A DONNER voir la notification de transmission du rapport d'examen préliminaire international (formulaire PCT/IPEA/416)	
Demande internationale n° PCT/FR00/02349	Date du dépôt international (jour/mois/année) 21/08/2000	Date de priorité (jour/mois/année) 23/08/1999
Classification internationale des brevets (CIB) ou à la fois classification nationale et CIB G06F9/00		
Déposant TRUSTED LOGIC et al.		

1. Le présent rapport d'examen préliminaire international, établi par l'administration chargée de l'examen préliminaire international, est transmis au déposant conformément à l'article 36.


2. Ce RAPPORT comprend 5 feuilles, y compris la présente feuille de couverture.

☐ Il est accompagné d'ANNEXES, c'est-à-dire de feuilles de la description, des revendications ou des dessins qui ont été modifiées et qui servent de base au présent rapport ou de feuilles contenant des rectifications faites auprès de l'administration chargée de l'examen préliminaire international (voir la règle 70.16 et l'instruction 607 des Instructions administratives du PCT).

Ces annexes comprennent feuilles.

3. Le présent rapport contient des indications relatives aux points suivants:

- I ☒ Base du rapport
- II ☐ Priorité
- III ☐ Absence de formulation d'opinion quant à la nouveauté, l'activité inventive et la possibilité d'application industrielle
- IV ☐ Absence d'unité de l'invention
- V ☒ Déclaration motivée selon l'article 35(2) quant à la nouveauté, l'activité inventive et la possibilité d'application industrielle; citations et explications à l'appui de cette déclaration
- VI ☐ Certains documents cités
- VII ☐ Irrégularités dans la demande internationale
- VIII ☒ Observations relatives à la demande internationale

Date de présentation de la demande d'examen préliminaire internationale 20/03/2001	Date d'achèvement du présent rapport 20.11.2001
Nom et adresse postale de l'administration chargée de l'examen préliminaire international:  Office européen des brevets D-80298 Munich Tél. +49 89 2399 - 0 Tx: 523656 epmu d Fax: +49 89 2399 - 4465	Fonctionnaire autorisé Lacroix, P N° de téléphone +49 89 2399 2707



THIS PAGE BLANK (U)

RAPPORT D'EXAMEN PRÉLIMINAIRE INTERNATIONAL

Demande internationale n° PCT/FR00/02349

I. Base du rapport

1. En ce qui concerne les **éléments** de la demande internationale (*les feuilles de remplacement qui ont été remises à l'office récepteur en réponse à une invitation faite conformément à l'article 14 sont considérées dans le présent rapport comme "initialement déposées" et ne sont pas jointes en annexe au rapport puisqu'elles ne contiennent pas de modifications (règles 70.16 et 70.17)*):

Description, pages:

1-68 version initiale

Revendications, N°:

1-27 version initiale

Dessins, feuilles:

1-14 version initiale

2. En ce qui concerne la **langue**, tous les éléments indiqués ci-dessus étaient à la disposition de l'administration ou lui ont été remis dans la langue dans laquelle la demande internationale a été déposée, sauf indication contraire donnée sous ce point.

Ces éléments étaient à la disposition de l'administration ou lui ont été remis dans la langue suivante: , qui est :

- ☐ la langue d'une traduction remise aux fins de la recherche internationale (selon la règle 23.1(b)).
- ☐ la langue de publication de la demande internationale (selon la règle 48.3(b)).
- ☐ la langue de la traduction remise aux fins de l'examen préliminaire internationale (selon la règle 55.2 ou 55.3).

3. En ce qui concerne les **séquences de nucléotides ou d'acide aminés** divulguées dans la demande internationale (le cas échéant), l'examen préliminaire internationale a été effectué sur la base du listage des séquences :

- ☐ contenu dans la demande internationale, sous forme écrite.
- ☐ déposé avec la demande internationale, sous forme déchiffrable par ordinateur.
- ☐ remis ultérieurement à l'administration, sous forme écrite.
- ☐ remis ultérieurement à l'administration, sous forme déchiffrable par ordinateur.
- ☐ La déclaration, selon laquelle le listage des séquences par écrit et fourni ultérieurement ne va pas au-delà de la divulgation faite dans la demande telle que déposée, a été fournie.
- ☐ La déclaration, selon laquelle les informations enregistrées sous déchiffrable par ordinateur sont identiques à celles du listage des séquences Présenté par écrit, a été fournie.

4. Les modifications ont entraîné l'annulation :

THIS PAGE BLANK (USPTO)

**RAPPORT D'EXAMEN
PRÉLIMINAIRE INTERNATIONAL**

Demande internationale n° PCT/FR00/02349

- ☐ de la description, pages :
- ☐ des revendications, n°s :
- ☐ des dessins, feuilles :

5. ☐ Le présent rapport a été formulé abstraction faite (de certaines) des modifications, qui ont été considérées comme allant au-delà de l'exposé de l'invention tel qu'il a été déposé, comme il est indiqué ci-après (règle 70.2(c)) :

(Toute feuille de remplacement comportant des modifications de cette nature doit être indiquée au point 1 et annexée au présent rapport)

6. Observations complémentaires, le cas échéant :

V. Déclaration motivée selon l'article 35(2) quant à la nouveauté, l'activité inventive et la possibilité d'application industrielle; citations et explications à l'appui de cette déclaration

1. Déclaration

Nouveauté	Oui : Revendications 1-27
	Non : Revendications
Activité inventive	Oui : Revendications 1-27
	Non : Revendications
Possibilité d'application industrielle	Oui : Revendications 1-27
	Non : Revendications

**2. Citations et explications
voir feuille séparée**

VIII. Observations relatives à la demande internationale

Les observations suivantes sont faites au sujet de la clarté des revendications, de la description et des dessins et de la question de savoir si les revendications se fondent entièrement sur la description :
voir feuille séparée

THIS PAGE BLANK (ISPTO)

Concernant le point V

Déclaration motivée selon l'article 35(2) quant à la nouveauté, l'activité inventive et la possibilité d'application industrielle; citations et explications à l'appui de cette déclaration

- 1). Il est fait référence au/x/ document/s/ suivant/s/:

D1: US - A - 5 748 964

- 2). La présente invention concerne un procédé de transformation d'un code objet classique constituant d'une applique permettant une exécution par un système informatique embarqué disposant de faibles ressources.

Etat de la Technique:

Le téléchargement d'une applique sur un système informatique embarqué est soumis à une vérification d'authenticité. D1 divulgue une vérification statique simulant l'exécution de l'applique au niveau des types de données et établit, une fois pour toutes, que le code de l'applique respecte les règles des types de données et de contrôle d'accès imposée par la machine virtuelle et ne provoque pas de débordement de pile.

Problème:

Cette solution présente l'inconvénient d'un processus de vérification statique du code complexe et coûteux, tant en taille de code nécessaire pour contrôler le processeur, qu'en taille de mémoire vive nécessaire pour contenir les résultats intermédiaires de la vérification, et qu'en temps de calcul. Ces besoins en mémoire dépassent largement les capacités des ressources de la majorité des systèmes informatiques embarqués actuels.

Solution:

La présente invention met en oeuvre un processus de normalisation d'un code objet d'origine en un code objet normalisé à instruction de branchement à pile vide et en un code normalisé faisant appel à des registres typés tels qu'un même registre est utilisé sous un seul et même type dans tout le code d'un sous-programme. Contrairement aux procédés de l'art antérieur, dans lesquels il est

THIS PAGE BLANK (USPTO)

nécessaire de conserver en mémoire le type de la pile à chaque cible de branchement, le procédé de vérification objet de la présente invention, n'a besoin que du type de la pile d'exécution lors de l'instruction en cours de vérification et il ne conserve pas en mémoire le type de cette pile pour d'autres sous-programmes. Il s'ensuit un besoin de capacité de mémoire réduit.

- 3). Les revendications dépendantes ont pour objet des formes particulières de mise en oeuvre de l'invention selon les revendications indépendantes. Elles satisfont donc également aux critères de nouveauté, d'activité inventive et d'application industrielle.

Concernant le point VIII

Observations relatives à la demande internationale

Il ressort clairement de la page 38 de la description que la caractéristique selon laquelle la pile doit être vide à chaque instruction de branchement ou de cible de branchement, et/ou que tous les registres lors de l'initialisation de la méthode soient initialisés à zéro est essentielle à la définition de l'invention. En effet, la simple phrase "une actualisation de l'effet de ladite instruction courante sur la pile des types et sur le tableau des types de registres n'implique en aucune façon que les conditions C3, C4 (page 38) à la base de la présente invention soient remplies.

Les revendications indépendantes ne contenant pas ces caractéristiques, elles ne remplissent pas les dispositions visées à l'article 6 PCT en combinaison avec la règle 6.3 b) PCT, qui prévoient qu'une revendication indépendante doit contenir toutes les caractéristiques techniques essentielles à la définition de l'invention. Par conséquent, les revendications indépendantes ne satisfont pas aux conditions requises à l'article 6 PCT.

THIS PAGE BLANK (USPTO)

PCT

REQUÊTE

Le soussigné requiert que la présente demande internationale soit traitée conformément au Traité de coopération en matière de brevets.

Réservé à l

récepteur

Demande internationale n°

Date du dépôt international

Nom de l'office récepteur et "Demande internationale PCT"

 Référence du dossier du déposant ou du mandataire (facultatif)
 (12 caractères au maximum)

BCT000077

Cadre n° I TITRE DE L'INVENTION PROTOCOLE DE GESTION, PROCEDE DE VERIFICATION ET DE TRANSFORMATION D'UN FRAGMENT DE PROGRAMME TELECHARGE ET SYSTEMES CORRESPONDANTS

Cadre n° II DÉPOSANT

Nom et adresse : (Nom de famille suivi du prénom; pour une personne morale, désignation officielle complète. L'adresse doit comprendre le code postal et le nom du pays. Le pays de l'adresse indiquée dans ce cadre est l'Etat où le déposant a son domicile si aucun domicile n'est indiqué ci-dessous.)

TRUSTED LOGIC
 23, avenue de Fulpmes
 78450 VILLEPREUX
 FRANCE

☐ Cette personne est aussi inventeur.

n° de téléphone

n° de télécopieur

n° de téléimprimeur

Nationalité (nom de l'Etat) :

FR

Domicile (nom de l'Etat) :

FR

 Cette personne est
 déposant pour :

☐
tous les États
désignés
☒
tous les États désignés sauf
les États-Unis d'Amérique
☐
les États-Unis d'Amérique
seulement
☐
les États indiqués dans
le cadre supplémentaire

Cadre n° III AUTRE(S) DÉPOSANT(S) OU (AUTRE(S)) INVENTEUR(S)

Nom et adresse : (Nom de famille suivi du prénom; pour une personne morale, désignation officielle complète. L'adresse doit comprendre le code postal et le nom du pays. Le pays de l'adresse indiquée dans ce cadre est l'Etat où le déposant a son domicile si aucun domicile n'est indiqué ci-dessous.)

LEROY Xavier
 88bis, avenue de Paris
 78000 VERSAILLES
 FRANCE

Cette personne est :

☐ déposant seulement

☒ déposant et inventeur

☐ inventeur seulement
 (Si cette case est cochée,
 ne pas remplir la suite.)

Nationalité (nom de l'Etat) :

FR

Domicile (nom de l'Etat) :

FR

 Cette personne est
 déposant pour :

☐
tous les États
désignés
☐
tous les États désignés sauf
les États-Unis d'Amérique
☒
les États-Unis d'Amérique
seulement
☐
les États indiqués dans
le cadre supplémentaire
☐ D'autres déposants ou inventeurs sont indiqués sur une feuille annexe.

Cadre n° IV MANDATAIRE OU REPRÉSENTANT COMMUN; OU ADRESSE POUR LA CORRESPONDANCE

La personne dont l'identité est donnée ci-dessous est/à été désignée pour agir au nom du ou des déposants auprès des autorités internationales compétentes, comme:

☒

mandataire

☐

représentant commun

Nom et adresse : (Nom de famille suivi du prénom; pour une personne morale, désignation officielle complète. L'adresse doit comprendre le code postal et le nom du pays.)

FRECHEDE Michel - FORT Jacques - JACQUELIN
 Marc-Henri - LOISEL Bertrand - VERDURE Stéphane
 CABINET PLASSERAUD
 84 rue d'Amsterdam
 75440 PARIS CEDEX 09
 FRANCE

n° de téléphone

01 44 63 41 11

n° de télécopieur

01 42 80 01 59

n° de téléimprimeur

☐ Adresse pour la correspondance : cocher cette case lorsque aucun mandataire ni représentant commun n'est/n'a été désigné et que l'espace ci-dessus est utilisé pour indiquer une adresse spéciale à laquelle la correspondance doit être envoyée.

THIS PAGE BLANK (USPTO)

Cadre n° V DESIGNATION D'ETATS

Les désignations suivantes sont faites conformément à la règle 4.9.a) (cocher les cases appropriées, une au moins doit l'être) :

Brevet régional

- ☐ AP Brevet ARIPO : GH Ghana, GM Gambie, KE Kenya, LS Lesotho, MW Malawi, MZ Mozambique, SD Soudan, SL Sierra Leone, SZ Swaziland, TZ République-Unie de Tanzanie, UG Ouganda, ZW Zimbabwe et tout autre Etat qui est un Etat contractant du Protocole de Harare et du PCT
- ☐ EA Brevet eurasien : AM Arménie, AZ Azerbaïdjan, BY Bélarus, KG Kirghizistan, KZ Kazakhstan, MD République de Moldova, RU Fédération de Russie, TJ Tadjikistan, TM Turkménistan et tout autre Etat qui est un Etat contractant de la Convention sur le brevet eurasien et du PCT
- ☒ EP Brevet européen : AT Autriche, BE Belgique, CH et LI Suisse et Liechtenstein, CY Chypre, DE Allemagne, DK Danemark, ES Espagne, FI Finlande, FR France, GB Royaume-Uni, GR Grèce, IE Irlande, IT Italie, LU Luxembourg, MC Monaco, NL Pays-Bas, PT Portugal, SE Suède et tout autre Etat qui est un Etat contractant de la Convention sur le brevet européen et du PCT
- ☐ OA Brevet OAPI : BF Burkina Faso, BJ Bénin, CF République centrafricaine, CG Congo, CI Côte d'Ivoire, CM Cameroun, GA Gabon, GN Guinée, GW Guinée-Bissau, ML Mali, MR Mauritanie, NE Niger, SN Sénégal, TD Tchad, TG Togo et tout autre Etat qui est un Etat membre de l'OAPI et un Etat contractant du PCT (si une autre forme de protection ou de traitement est souhaitée, le préciser sur la ligne pointillée) :

Brevet national (si une autre forme de protection ou de traitement est souhaitée, le préciser sur la ligne pointillée) :

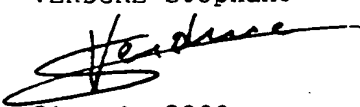
- | | |
|--|---|
| <input type="checkbox"/> AE Emirats arabes unis | <input type="checkbox"/> LC Sainte-Lucie |
| <input type="checkbox"/> AG Antigua-et-Barbuda | <input type="checkbox"/> LK Sri Lanka |
| <input type="checkbox"/> AL Albanie | <input type="checkbox"/> LR Liberia |
| <input type="checkbox"/> AM Arménie | <input type="checkbox"/> LS Lesotho |
| <input type="checkbox"/> AT Autriche | <input type="checkbox"/> LT Lituanie |
| <input checked="" type="checkbox"/> AU Australie | <input type="checkbox"/> LU Luxembourg |
| <input type="checkbox"/> AZ Azerbaïdjan | <input type="checkbox"/> LV Lettonie |
| <input type="checkbox"/> BA Bosnie-Herzégovine | <input type="checkbox"/> MA Maroc |
| <input type="checkbox"/> BB Barbade | <input type="checkbox"/> MD République de Moldova |
| <input type="checkbox"/> BG Bulgarie | <input type="checkbox"/> MG Madagascar |
| <input type="checkbox"/> BR Brésil | <input type="checkbox"/> MK Ex-République yougoslave de Macédoine |
| <input type="checkbox"/> BY Bélarus | <input type="checkbox"/> MN Mongolie |
| <input type="checkbox"/> BZ Belize | <input type="checkbox"/> MW Malawi |
| <input checked="" type="checkbox"/> CA Canada | <input type="checkbox"/> MX Mexique |
| <input type="checkbox"/> CH et LI Suisse et Liechtenstein | <input type="checkbox"/> MZ Mozambique |
| <input checked="" type="checkbox"/> CN Chine | <input type="checkbox"/> NO Norvège |
| <input type="checkbox"/> CR Costa Rica | <input type="checkbox"/> NZ Nouvelle-Zélande |
| <input type="checkbox"/> CU Cuba | <input type="checkbox"/> PL Pologne |
| <input type="checkbox"/> CZ République tchèque | <input type="checkbox"/> PT Portugal |
| <input type="checkbox"/> DE Allemagne | <input type="checkbox"/> RO Roumanie |
| <input type="checkbox"/> DK Danemark | <input type="checkbox"/> RU Fédération de Russie |
| <input type="checkbox"/> DM Dominique | <input type="checkbox"/> SD Soudan |
| <input type="checkbox"/> DZ Algérie | <input type="checkbox"/> SE Suède |
| <input type="checkbox"/> EE Estonie | <input type="checkbox"/> SG Singapour |
| <input type="checkbox"/> ES Espagne | <input type="checkbox"/> SI Slovénie |
| <input type="checkbox"/> FI Finlande | <input type="checkbox"/> SK Slovaquie |
| <input type="checkbox"/> GB Royaume-Uni | <input type="checkbox"/> SL Sierra Leone |
| <input type="checkbox"/> GD Grenade | <input type="checkbox"/> TJ Tadjikistan |
| <input type="checkbox"/> GE Géorgie | <input type="checkbox"/> TM Turkménistan |
| <input type="checkbox"/> GH Ghana | <input type="checkbox"/> TR Turquie |
| <input type="checkbox"/> GM Gambie | <input type="checkbox"/> TT Trinité-et-Tobago |
| <input type="checkbox"/> HR Croatie | <input type="checkbox"/> TZ République-Unie de Tanzanie |
| <input type="checkbox"/> HU Hongrie | <input type="checkbox"/> UA Ukraine |
| <input type="checkbox"/> ID Indonésie | <input type="checkbox"/> UG Ouganda |
| <input type="checkbox"/> IL Israël | <input checked="" type="checkbox"/> US États-Unis d'Amérique |
| <input type="checkbox"/> IN Inde | <input type="checkbox"/> UZ Ouzbékistan |
| <input type="checkbox"/> IS Islande | <input type="checkbox"/> VN Viet Nam |
| <input checked="" type="checkbox"/> JP Japon | <input type="checkbox"/> YU Yougoslavie |
| <input type="checkbox"/> KE Kenya | <input type="checkbox"/> ZA Afrique du Sud |
| <input type="checkbox"/> KG Kirghizistan | <input type="checkbox"/> ZW Zimbabwe |
| <input type="checkbox"/> KP République populaire démocratique de Corée | |
| <input type="checkbox"/> KR République de Corée | |
| <input type="checkbox"/> KZ Kazakhstan | |

Case réservée pour la désignation d'États qui sont devenus parties au PCT après la publication de la présente feuille :

☐

Déclaration concernant les désignations de précaution : outre les désignations faites ci-dessus, le déposant fait aussi conformément à la règle 4.9.b) toutes les désignations qui seraient autorisées en vertu du PCT, à l'exception de toute désignation indiquée dans le cadre supplémentaire comme étant exclue de la portée de cette déclaration. Le déposant déclare que ces désignations additionnelles sont faites sous réserve de confirmation et que toute désignation qui n'est pas confirmée avant l'expiration d'un délai de 15 mois à compter de la date de priorité doit être considérée comme retirée par le déposant à l'expiration de ce délai. (La confirmation (y compris les taxes) doit parvenir à l'office récepteur dans le délai de 15 mois.)

THIS PAGE BLANK (USPTO)

Cadre n° VI REVENDEICATION DE PRIORITÉ		<input type="checkbox"/> D'autres revendications de priorité sont indiquées dans le cadre supplémentaire.		
Date de dépôt de la demande antérieure (jour/mois/année)	Numéro de la demande antérieure	Lorsque la demande antérieure est une :		
		demande nationale : pays	demande régionale : * office régional	demande internationale : office récepteur
(1) 23/08/1999	99 10697	FRANCE		
(2)				
(3)				
<input type="checkbox"/> L'office récepteur est prié de préparer et de transmettre au Bureau international une copie certifiée conforme de la ou des demandes antérieures (seulement si la demande antérieure a été déposée auprès de l'office qui, aux fins de la présente demande internationale, est l'office récepteur) indiquées ci-dessus au(x) point(s) : _____				
* Si la demande antérieure est une demande ARIPO, il est obligatoire d'indiquer dans le cadre supplémentaire au moins un pays partie à la Convention de Paris pour la protection de la propriété industrielle pour lequel cette demande antérieure a été déposée (règle 4.10.b)ii). Voir le cadre supplémentaire.				
Cadre n° VII ADMINISTRATION CHARGÉE DE LA RECHERCHE INTERNATIONALE				
Choix de l'administration chargée de la recherche internationale (ISA) (si plusieurs administrations chargées de la recherche internationale sont compétentes pour procéder à la recherche internationale, indiquer l'administration choisie; le code à deux lettres peut être utilisé) : ISA / EP		Demande d'utilisation des résultats d'une recherche antérieure; mention de cette recherche (si une recherche antérieure a été effectuée par l'administration chargée de la recherche internationale ou demandée à cette dernière) : Date (jour/mois/année) Numéro Pays (ou office régional) 04/07/2000 FA 583732 FRANCE		
Cadre n° VIII BORDEREAU; LANGUE DE DÉPÔT				
La présente demande internationale contient le nombre de feuilles suivant : requête : 3 description (sauf partie réservée au listage des séquences) : 68 revendications : 17 abrégé : 1 dessins : 14 partie de la description réservée au listage des séquences : _____ Nombre total de feuilles : 103		Le ou les éléments cochés ci-après sont joints à la présente demande internationale : 1. <input checked="" type="checkbox"/> feuille de calcul des taxes 2. <input type="checkbox"/> pouvoir distinct signé 3. <input type="checkbox"/> copie du pouvoir général; numéro de référence, le cas échéant : 4. <input type="checkbox"/> explication de l'absence d'une signature 5. <input type="checkbox"/> document(s) de priorité indiqué(s) dans le cadre n° VI au(x) point(s) : 6. <input type="checkbox"/> traduction de la demande internationale en (langue) : 7. <input type="checkbox"/> indications séparées concernant des micro-organismes ou autre matériel biologique déposés 8. <input type="checkbox"/> listage des séquences de nucléotides ou d'acides aminés sous forme déchiffrable par ordinateur 9. <input checked="" type="checkbox"/> autres éléments (préciser) : rapport de recherche français		
Figure des dessins qui doit accompagner l'abrégé : 2		Langue de dépôt de la demande internationale : FRANCAIS		
Cadre n° IX SIGNATURE DU DÉPOSANT OU DU MANDATAIRE				
À côté de chaque signature, indiquer le nom du signataire et, si cela n'apparaît pas clairement à la lecture de la requête, à quel titre l'intéressé signe.				
VERDURE Stéphane  21 août 2000				

Réservé à l'office récepteur	
1. Date effective de réception des pièces supposées constituer la demande internationale : 3. Date effective de réception, rectifiée en raison de la réception ultérieure, mais dans les délais, de documents ou de dessins complétant ce qui est supposé constituer la demande internationale : 4. Date de réception, dans les délais, des corrections demandées selon l'article 11.2) du PCT :	2. Dessins : <input type="checkbox"/> reçus : <input type="checkbox"/> non reçus :
5. Administration chargée de la recherche internationale (si plusieurs sont compétentes) : ISA /	6. <input type="checkbox"/> Transmission de la copie de recherche différée jusqu'au paiement de la taxe de recherche.

Réservé au Bureau international	
Date de réception de l'exemplaire original par le Bureau international :	

THIS PAGE BLANK (USPTO)

PCT

FEUILLE DE CALCUL DES TAXES

Annexe de la requête

Réserve à l'office récepteur

Demande internationale n°

Référence du dossier du
déposant ou du mandataire BCT000077

Timbre à date de l'office récepteur

Déposant
TRUSTED LOGIC

CALCUL DES TAXES PRESCRITES

1. TAXE DE TRANSMISSION

400,00 FF T

2. TAXE DE RECHERCHE

6 198,79 FF S

Recherche internationale à effectuer par

(Si plusieurs administrations chargées de la recherche internationale sont compétentes en ce qui concerne la demande internationale, inscrire le nom de celle qui est choisie pour la recherche internationale.)

3. TAXE INTERNATIONALE

Taxe de base

La demande internationale contient 103 feuilles.

30 premières feuilles

2 682,86 FF b1

73

x 59,94 FF

= 4 309,92 FF b2

feuilles suivantes montant additionnel

Additionner les montants portés dans les cadres

b1 et b2 et inscrire le total dans le cadre B

6 992,78 FF B

Taxes de désignation

La demande internationale contient 6 désignations.

6

x 577,24 FF

= 3 463,44 FF D

nombre de taxes de
désignation dues (maximum 8) montant de la taxe de désignation

Additionner les montants portés dans les cadres B et D, et
inscrire le total dans le cadre I

(Les déposants de certains États ont droit à une réduction de 75 % sur la taxe internationale. Lorsque le déposant a (ou tous les déposants ont) droit à cette réduction, la somme devant figurer sous I est égale à 25 % de la somme des montants figurant sous B et D.)

10 456,22 FF I

4. TAXE AFFÉRENTE AU DOCUMENT DE PRIORITÉ (le cas échéant)

P

5. TOTAL DES TAXES DUES

Additionner les montants portés dans les cadres

T, S, I et P, et inscrire le résultat dans le cadre TOTAL.

17 055,01 FF

TOTAL

☐ Les taxes de désignation seront payées ultérieurement.

MODE DE PAIEMENT

☐ autorisation de débiter un compte
de dépôt (voir ci-dessous)

☐ traite bancaire

☐ coupons

☒ chèque

☐ espèces

☐ autres (préciser):

☐ mandat postal

☐ timbres fiscaux

AUTORISATION CONCERNANT UN COMPTE DE DÉPÔT (les offices récepteurs ne permettent pas tous l'utilisation de ce mode de paiement)

L'office récepteur/ ☐ est autorisé à débiter mon compte de dépôt du total des taxes indiqué ci-dessus.

☐ (cette case ne peut être cochée que si les conditions relatives aux comptes de dépôt établies par l'office récepteur le permettent) est autorisé à débiter mon compte de dépôt de tout montant manquant – ou à le créditer de tout excédent – dans le paiement du total des taxes indiqué ci-dessus.

☐ est autorisé à débiter mon compte de dépôt du montant de la taxe afférente à l'établissement du document de priorité et à sa transmission au Bureau international de l'OMPI.

Numéro du compte de dépôt

Date (jour/mois/année)

Signature

THIS PAGE BLANK (USPTO)

INTERNATIONAL SEARCH REPORT

Inter Application No

PCT/FK 00/02349

A. CLASSIFICATION OF SUBJECT MATTER
 IPC 7 G06F9/445 G06F9/45

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

IPC 7 G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)

EPO-Internal, PAJ, INSPEC, COMPENDEX

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	US 5 740 441 A (GOSLING JAMES A ET AL) 14 April 1998 (1998-04-14)	1-5, 7, 8, 10-13, 19, 20, 23, 24
A	column 1, line 1 - column 15, line 11	15, 16, 21, 25
X	column 17, line 7 - line 10 column 18, line 21 - line 23 column 18, line 32 - line 34	26, 27
Y	GOSLING J ET AL: "THE JAVATM LANGUAGE SPECIFICATION", JAVA LANGUAGE SPECIFICATION, XX, XX, PAGE(S) 215-236 XP002042923	1-5, 7, 8, 10-13, 19, 20, 23, 24
A	page 218, line 10 - page 220, line 35 --- -/-	6, 9



Further documents are listed in the continuation of box C.



Patent family members are listed in annex.

* Special categories of cited documents:

- *A* document defining the general state of the art which is not considered to be of particular relevance
- *E* earlier document but published on or after the international filing date
- *L* document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
- *O* document referring to an oral disclosure, use, exhibition or other means
- *P* document published prior to the international filing date but later than the priority date claimed

T later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

X document of particular relevance: the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

Y document of particular relevance: the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

G document member of the same patent family

Date of the actual completion of the international search

11 June 2001

Date of mailing of the international search report

19/06/2001

Name and mailing address of the ISA

European Patent Office, P.B. 5818 Patentlaan 2
 NL - 2280 HV Rijswijk
 Tel. (+31-70) 340-2040, Tx. 31 651 epo nl.
 Fax: (+31-70) 340-3016

Authorized officer

Kingma, Y

INTERNATIONAL SEARCH REPORT

Application No
PCT/FR 00/02349

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	GONG L ET AL: "Going beyond the sandbox: an overview of the new security architecture in the JavaDevelopment Kit 1.2" , PROCEEDINGS OF THE USENIX SYMPOSIUM ON INTERNET TECHNOLOGIES AND SYSTEMS XP002100907 page 107, right-hand column, line 1 -page 108, left-hand column, line 3 -----	3
A	GUTHERY: "JAVA CARD: Internet Computing on a Smart Card" IEEE INTERNET COMPUTING,US,IEEE SERVICE CENTER, PISCATAWAY, NJ, 1 February 1997 (1997-02-01), pages 57-59, XP002077647 ISSN: 1089-7801 page 58, right-hand column, line 19 -page 59, middle column, line 24 -----	1,4,19, 26

INTERNATIONAL SEARCH REPORT

Information on patent family members

International Application No

PCT/FR 00/02349

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 5740441 A	14-04-1998	US 5748964 A	05-05-1998
		US 5999731 A	07-12-1999
		EP 0718764 A	26-06-1996
		JP 8234994 A	13-09-1996
<hr/>			

This Page Blank (uspto)

RAPPORT DE RECHERCHE INTERNATIONALE

Dem. internationale No
PCT/FR 00/02349

A. CLASSEMENT DE L'OBJET DE LA DEMANDE
CIB 7 G06F9/445 G06F9/45

Selon la classification internationale des brevets (CIB) ou à la fois selon la classification nationale et la CIB

B. DOMAINES SUR LESQUELS LA RECHERCHE A PORTE

Documentation minimale consultée (système de classification suivi des symboles de classement)

CIB 7 G06F

Documentation consultée autre que la documentation minimale dans la mesure où ces documents relèvent des domaines sur lesquels a porté la recherche

Base de données électronique consultée au cours de la recherche internationale (nom de la base de données, et si réalisable, termes de recherche utilisés)

EPO-Internal, PAJ, INSPEC, COMPENDEX

C. DOCUMENTS CONSIDERES COMME PERTINENTS

Catégorie *	Identification des documents cités, avec, le cas échéant, l'indication des passages pertinents	no. des revendications visées
Y	US 5 740 441 A (GOSLING JAMES A ET AL) 14 avril 1998 (1998-04-14)	1-5, 7, 8, 10-13, 19, 20, 23, 24
A	colonne 1, ligne 1 - colonne 15, ligne 11	15, 16, 21, 25
X	colonne 17, ligne 7 - ligne 10 colonne 18, ligne 21 - ligne 23 colonne 18, ligne 32 - ligne 34	26, 27
Y	GOSLING J ET AL: "THE JAVATM LANGUAGE SPECIFICATION", JAVA LANGUAGE SPECIFICATION, XX, XX, PAGE(S) 215-236 XP002042923	1-5, 7, 8, 10-13, 19, 20, 23, 24
A	page 218, ligne 10 - page 220, ligne 35	6, 9
	--- -/--	

☒ Voir la suite du cadre C pour la fin de la liste des documents

☒ Les documents de familles de brevets sont indiqués en annexe

* Catégories spéciales de documents cités:

- *A* document définissant l'état général de la technique, non considéré comme particulièrement pertinent
- *E* document antérieur, mais publié à la date de dépôt international ou après cette date
- *L* document pouvant jeter un doute sur une revendication de priorité ou cité pour déterminer la date de publication d'une autre citation ou pour une raison spéciale (telle qu'indiquée)
- *O* document se référant à une divulgation orale, à un usage, à une exposition ou tous autres moyens
- *P* document publié avant la date de dépôt international, mais postérieurement à la date de priorité revendiquée

- *T* document ultérieur publié après la date de dépôt international ou la date de priorité et n'appartenant pas à l'état de la technique pertinent, mais cité pour comprendre le principe ou la théorie constituant la base de l'invention
- *X* document particulièrement pertinent: l'invention revendiquée ne peut être considérée comme nouvelle ou comme impliquant une activité inventive par rapport au document considéré isolément
- *Y* document particulièrement pertinent: l'invention revendiquée ne peut être considérée comme impliquant une activité inventive lorsque le document est associé à un ou plusieurs autres documents de même nature, cette combinaison étant évidente pour une personne du métier
- *Z* document qui fait partie de la même famille de brevets

Date à laquelle la recherche internationale a été effectivement achevée

11 juin 2001

Date d'expédition du présent rapport de recherche internationale

19/06/2001

Nom et adresse postale de l'administration chargée de la recherche internationale
Office Européen des Brevets, P.B. 5818 Patentlaan 2
NL - 2280 HV Rijswijk
Tel. (+31-70) 340-2040, Tx. 31 651 epo nl,
Fax: (+31-70) 340-3016

Fonctionnaire autorisé

Kingma, Y

RAPPORT DE RECHERCHE INTERNATIONALE

Document Internationale No
PCT/FR 00/02349

C.(suite) DOCUMENTS CONSIDERES COMME PERTINENTS

Categorie	Identification des documents cités, avec, le cas échéant, l'indication des passages pertinents	no. des revendications visées
A	<p>GONG L ET AL: "Going beyond the sandbox: an overview of the new security architecture in the JavaDevelopment Kit 1.2" , PROCEEDINGS OF THE USENIX SYMPOSIUM ON INTERNET TECHNOLOGIES AND SYSTEMS XP002100907 page 107, colonne de droite, ligne 1 -page 108, colonne de gauche, ligne 3</p>	3
A	<p>GUTHERY: "JAVA CARD: Internet Computing on a Smart Card" IEEE INTERNET COMPUTING,US,IEEE SERVICE CENTER, PISCATAWAY, NJ, 1 février 1997 (1997-02-01), pages 57-59, XP002077647 ISSN: 1089-7801 page 58, colonne de droite, ligne 19 -page 59, colonne du milieu, ligne 24</p>	1,4,19, 26

RAPPORT DE RECHERCHE INTERNATIONALE

Renseignements relatifs aux familles de brevets

Demande internationale No

PCT/FK 00/02349

Document brevet cité au rapport de recherche	Date de publication	Membre(s) de la famille de brevet(s)	Date de publication
US 5740441 A	14-04-1998	US 5748964 A	05-05-1998
		US 5999731 A	07-12-1999
		EP 0718764 A	26-06-1996
		JP 8234994 A	13-09-1996

THIS PAGE BLANK (USPTO)

TRAITE DE COOPERATION EN MATIERE DE BREVETS

PCT

REC'D 22 NOV 2001

WIPO PCT

RAPPORT D'EXAMEN PRELIMINAIRE INTERNATIONAL

(article 36 et règle 70 du PCT)

Référence du dossier du déposant ou du mandataire BCT000077	POUR SUITE A DONNER voir la notification de transmission du rapport d'examen préliminaire international (formulaire PCT/IPEA/416)	
Demande internationale n° PCT/FR00/02349	Date du dépôt international (jour/mois/année) 21/08/2000	Date de priorité (jour/mois/année) 23/08/1999
Classification internationale des brevets (CIB) ou à la fois classification nationale et CIB G06F9/00		
Déposant TRUSTED LOGIC et al.		

1. Le présent rapport d'examen préliminaire international, établi par l'administration chargée de l'examen préliminaire international, est transmis au déposant conformément à l'article 36.


2. Ce RAPPORT comprend 5 feuilles, y compris la présente feuille de couverture.

☐ Il est accompagné d'ANNEXES, c'est-à-dire de feuilles de la description, des revendications ou des dessins qui ont été modifiées et qui servent de base au présent rapport ou de feuilles contenant des rectifications faites auprès de l'administration chargée de l'examen préliminaire international (voir la règle 70.16 et l'instruction 607 des Instructions administratives du PCT).

Ces annexes comprennent feuilles.

3. Le présent rapport contient des indications relatives aux points suivants:

- I ☒ Base du rapport
- II ☐ Priorité
- III ☐ Absence de formulation d'opinion quant à la nouveauté, l'activité inventive et la possibilité d'application industrielle
- IV ☐ Absence d'unité de l'invention
- V ☒ Déclaration motivée selon l'article 35(2) quant à la nouveauté, l'activité inventive et la possibilité d'application industrielle; citations et explications à l'appui de cette déclaration
- VI ☐ Certains documents cités
- VII ☐ Irrégularités dans la demande internationale
- VIII ☒ Observations relatives à la demande internationale

Date de présentation de la demande d'examen préliminaire internationale 20/03/2001	Date d'achèvement du présent rapport 20.11.2001
Nom et adresse postale de l'administration chargée de l'examen préliminaire international:  Office européen des brevets D-80298 Munich Tél. +49 89 2399 - 0 Tx: 523656 epmu d Fax: +49 89 2399 - 4465	Fonctionnaire autorisé Lacroix, P N° de téléphone +49 89 2399 2707



THIS PAGE BLANK (USPTO)

RAPPORT D'EXAMEN PRÉLIMINAIRE INTERNATIONAL

Demande internationale n° PCT/FR00/02349

I. Bas du rapport

1. En ce qui concerne les **éléments** de la demande internationale (*les feuilles de remplacement qui ont été remises à l'office récepteur en réponse à une invitation faite conformément à l'article 14 sont considérées dans le présent rapport comme "initialement déposées" et ne sont pas jointes en annexe au rapport puisqu'elles ne contiennent pas de modifications (règles 70.16 et 70.17)*):

Description, pages:

1-68 version initiale

Revendications, N°:

1-27 version initiale

Dessins, feuilles:

1-14 version initiale

2. En ce qui concerne la **langue**, tous les éléments indiqués ci-dessus étaient à la disposition de l'administration ou lui ont été remis dans la langue dans laquelle la demande internationale a été déposée, sauf indication contraire donnée sous ce point.

Ces éléments étaient à la disposition de l'administration ou lui ont été remis dans la langue suivante: , qui est :

- ☐ la langue d'une traduction remise aux fins de la recherche internationale (selon la règle 23.1(b)).
- ☐ la langue de publication de la demande internationale (selon la règle 48.3(b)).
- ☐ la langue de la traduction remise aux fins de l'examen préliminaire internationale (selon la règle 55.2 ou 55.3).

3. En ce qui concerne les **séquences de nucléotides ou d'acide aminés** divulguées dans la demande internationale (le cas échéant), l'examen préliminaire internationale a été effectué sur la base du listage des séquences :

- ☐ contenu dans la demande internationale, sous forme écrite.
- ☐ déposé avec la demande internationale, sous forme déchiffrable par ordinateur.
- ☐ remis ultérieurement à l'administration, sous forme écrite.
- ☐ remis ultérieurement à l'administration, sous forme déchiffrable par ordinateur.
- ☐ La déclaration, selon laquelle le listage des séquences par écrit et fourni ultérieurement ne va pas au-delà de la divulgation faite dans la demande telle que déposée, a été fournie.
- ☐ La déclaration, selon laquelle les informations enregistrées sous déchiffrable par ordinateur sont identiques à celles du listage des séquences Présenté par écrit, a été fournie.

4. Les modifications ont entraîné l'annulation :

THIS PAGE BLANK (USPTO)

**RAPPORT D'EXAMEN
PRÉLIMINAIRE INTERNATIONAL**

Demande internationale n° PCT/FR00/02349

- ☐ de la description, pages :
- ☐ des revendications, n°s :
- ☐ des dessins, feuilles :

5. ☐ Le présent rapport a été formulé abstraction faite (de certaines) des modifications, qui ont été considérées comme allant au-delà de l'exposé de l'invention tel qu'il a été déposé, comme il est indiqué ci-après (règle 70.2(c)) :

(Toute feuille de remplacement comportant des modifications de cette nature doit être indiquée au point 1 et annexée au présent rapport)

6. Observations complémentaires, le cas échéant :

V. Déclaration motivée selon l'article 35(2) quant à la nouveauté, l'activité inventive et la possibilité d'application industrielle; citations et explications à l'appui de cette déclaration

1. Déclaration

Nouveauté	Oui : Revendications 1-27
	Non : Revendications
Activité inventive	Oui : Revendications 1-27
	Non : Revendications
Possibilité d'application industrielle	Oui : Revendications 1-27
	Non : Revendications

2. Citations et explications
voir feuille séparée

VIII. Observations relatives à la demande internationale

Les observations suivantes sont faites au sujet de la clarté des revendications, de la description et des dessins et de la question de savoir si les revendications se fondent entièrement sur la description :
voir feuille séparée

THIS PAGE BLANK (USPTO)

Concernant le point V

Déclaration motivée selon l'article 35(2) quant à la nouveauté, l'activité inventive et la possibilité d'application industrielle; citations et explications à l'appui de cette déclaration

- 1). Il est fait référence au/x/ document/s/ suivant/s/:

D1: US - A - 5 748 964

- 2). La présente invention concerne un procédé de transformation d'un code objet classique constituant d'une applique permettant une exécution par un système informatique embarqué disposant de faibles ressources.

Etat de la Technique:

Le téléchargement d'une applique sur un système informatique embarqué est soumis à une vérification d'authenticité. D1 divulgue une vérification statique simulant l'exécution de l'applique au niveau des types de données et établit, une fois pour toutes, que le code de l'applique respecte les règles des types de données et de contrôle d'accès imposée par la machine virtuelle et ne provoque pas de débordement de pile.

Problème:

Cette solution présente l'inconvénient d'un processus de vérification statique du code complexe et coûteux, tant en taille de code nécessaire pour contrôler le processeur, qu'en taille de mémoire vive nécessaire pour contenir les résultats intermédiaires de la vérification, et qu'en temps de calcul. Ces besoins en mémoire dépassent largement les capacités des ressources de la majorité des systèmes informatiques embarqués actuels.

Solution:

La présente invention met en oeuvre un processus de normalisation d'un code objet d'origine en un code objet normalisé à instruction de branchement à pile vide et en un code normalisé faisant appel à des registres typés tels qu'un même registre est utilisé sous un seul et même type dans tout le code d'un sous-programme. Contrairement aux procédés de l'art antérieur, dans lesquels il est

THIS PAGE BLANK (USPTO)

nécessaire de conserver en mémoire le type de la pile à chaque cible de branchement, le procédé de vérification objet de la présente invention, n'a besoin que du type de la pile d'exécution lors de l'instruction en cours de vérification et il ne conserve pas en mémoire le type de cette pile pour d'autres sous-programmes. Il s'ensuit un besoin de capacité de mémoire réduit.

- 3). Les revendications dépendantes ont pour objet des formes particulières de mise en oeuvre de l'invention selon les revendications indépendantes. Elles satisfont donc également aux critères de nouveauté, d'activité inventive et d'application industrielle.

Concernant le point VIII

Observations relatives à la demande internationale

Il ressort clairement de la page 38 de la description que la caractéristique selon laquelle la pile doit être vide à chaque instruction de branchement ou de cible de branchement, et/ou que tous les registres lors de l'initialisation de la méthode soient initialisés à zéro est essentielle à la définition de l'invention. En effet, la simple phrase "une actualisation de l'effet de ladite instruction courante sur la pile des types et sur le tableau des types de registres n'implique en aucune façon que les conditions C3, C4 (page 38) à la base de la présente invention soient remplies.

Les revendications indépendantes ne contenant pas ces caractéristiques, elles ne remplissent pas les dispositions visées à l'article 6 PCT en combinaison avec la règle 6.3 b) PCT, qui prévoient qu'une revendication indépendante doit contenir toutes les caractéristiques techniques essentielles à la définition de l'invention. Par conséquent, les revendications indépendantes ne satisfont pas aux conditions requises à l'article 6 PCT.

THIS PAGE BLANK (USPTO)

(12) DEMANDE INTERNATIONALE PUBLIÉE EN VERTU DU TRAITÉ DE COOPÉRATION
EN MATIÈRE DE BREVETS (PCT)

(19) Organisation Mondiale de la Propriété
Intellectuelle
Bureau international



PTO/PC

22 FEB 2002

(43) Date de la publication internationale
1 mars 2001 (01.03.2001)

PCT

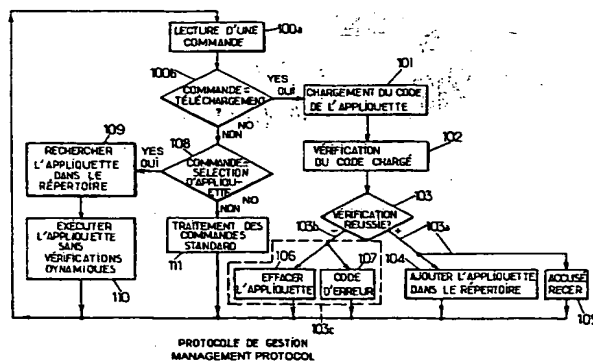
(10) Numéro de publication internationale
WO 01/14958 A2

- (51) Classification internationale des brevets⁷: G06F 9/00 (72) Inventeur; et
(75) Inventeur/Déposant (pour US seulement): LEROY, Xavier [FR/FR]; 88 bis, avenue de Paris, F-78000 Versailles (FR).
- (21) Numéro de la demande internationale:
PCT/FR00/02349
- (22) Date de dépôt international: 21 août 2000 (21.08.2000) (74) Mandataires: FRECHEDE, Michel etc.; Cabinet Plasseraud, 84, rue d'Amsterdam, F-75440 Paris Cedex 09 (FR).
- (25) Langue de dépôt: français
- (26) Langue de publication: français
- (81) États désignés (national): AU, CA, CN, JP, US.
- (30) Données relatives à la priorité:
99/10697 23 août 1999 (23.08.1999) FR (84) États désignés (régional): brevet européen (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE).
- (71) Déposant (pour tous les États désignés sauf US): Publiée:
TRUSTED LOGIC [FR/FR]; 23, avenue de Fulpmes, F-78450 Villepreux (FR). — Sans rapport de recherche internationale, sera republiée dès réception de ce rapport.

[Suite sur la page suivante]

(54) Title: MANAGEMENT PROTOCOL, METHOD FOR VERIFYING AND TRANSFORMING A DOWNLOADED PROGRAMME FRAGMENT AND CORRESPONDING SYSTEMS

(54) Titre: PROTOCOLE DE GESTION, PROCÉDE DE VERIFICATION ET DE TRANSFORMATION D'UN FRAGMENT DE PROGRAMME TELECHARGE ET SYSTEMES CORRESPONDANTS



109...SEARCH FOR APPLLET IN REPERTOIRE
110...EXECUTE APPLLET WITHOUT DYNAMIC CHECKS
100a...COMMANDE READ
100b...COMMANDE = DOWNLOAD ?
108...COMMANDE = SELECTION OF APPLLET
111...STANDARD COMMAND PROCESSING
101...APPLLET CODE DOWNLOADED
102...DOWNLOADED CODE VERIFIED
103...VERIFICATION SUCCESSFUL ?
106...DELETE APPLLET
107...ERROR CODE
104...ADD APPLLET TO REPERTOIRE
105...RECEIVE ACKNOWLEDGMENT

(57) Abstract: The invention relates to a management protocol and to a method for verifying a programme fragment, or applet, which has been downloaded onto a portable system. An applet downloading command (100a, 100b) is executed. Once a positive response has been received, the object code of the applet is read (101) and subjected (102) to a verification process, instruction by instruction. The verification process consists of a stage comprising the initialisation of the type stack and table of register types representing the state of the virtual machine of the portable system at the start of the execution of the applet code; and a verification, instruction by instruction, for each target current instruction, of the existence of a target branch instruction, a target exception handler call or a target sub-routine call, the effect of the instruction on the type stack and the table of register types being verified and updated. If the verification is successful (103a), the applet is registered (104) and an acknowledgement is sent (105) to the downloading drive. Otherwise, the applet is destroyed (106). The invention is suitable for use for portable systems in a Java environment.

[Suite sur la page suivante]



En ce qui concerne les codes à deux lettres et autres abréviations, se référer aux "Notes explicatives relatives aux codes et abréviations" figurant au début de chaque numéro ordinaire de la Gazette du PCT.

(57) Abrégé: L'invention concerne un protocole de gestion et un procédé de vérification d'un fragment de programme, ou applique, téléchargé sur un système embarqué. Une commande de téléchargement (100a, 100b) de l'applique est effectuée. Sur réponse positive, le code objet de l'applique est lu (101) et soumis (102) à une vérification instruction par instruction. La vérification consiste en une étape d'initialisation de la pile des types et du tableau des types de registres représentant l'état de la machine virtuelle du système embarqué au début de l'exécution du code de l'applique et en une vérification, instruction par instruction pour chaque instruction courante cible, de l'existence d'une cible d'instruction de branchement, d'appel d'un gestionnaire d'exceptions ou d'un appel de sous-routine, et par une vérification et une actualisation de l'effet de cette instruction sur la pile des types et le tableau des types de registres. Sur vérification réussie (103a), l'applique est enregistrée (104) et un accusé de réception est envoyé (105) au lecteur de téléchargement. L'applique est détruite (106) sinon. Application aux systèmes embarqués en environnement Java.

PROTOCOLE DE GESTION, PROCEDE DE VERIFICATION ET DE
TRANSFORMATION D'UN FRAGMENT DE PROGRAMME TELECHARGE ET
SYSTEMES CORRESPONDANTS

5 L'invention concerne un protocole de gestion, un
procédé de vérification, un procédé de transformation d'un
fragment de programme téléchargé et les systèmes
correspondants, plus particulièrement destinés aux
systèmes informatiques embarqués disposant de faibles
10 ressources en mémoire et en puissance de calcul.

D'une manière générale, en référence à la figure
1a, on rappelle que les systèmes informatiques embarqués
10 comportent un microprocesseur 11, une mémoire
permanente, telle qu'une mémoire non inscriptible 12
15 contenant le code du programme exécutable, une mémoire
permanente, non volatile, réinscriptible 13 de type EEPROM
contenant les données stockées dans le système, une
mémoire vive, volatile, 14 dans laquelle le programme
stocke ses résultats intermédiaires pendant son exécution,
20 et des dispositifs d'entrée/sortie 15 permettant au
système d'interagir avec son environnement. Dans le cas où
le système informatique embarqué est constitué par une
carte à microprocesseur, du type carte bancaire, le
dispositif d'entrée/sortie 15 consiste en une liaison
25 série permettant à la carte de communiquer avec un
terminal, tel qu'un terminal lecteur de cartes.

Dans les systèmes informatiques embarqués
classiques, le code du programme exécuté par le système
est figé lors de la construction du système, ou, au plus
30 tard, lors de la personnalisation de ce dernier avant
livraison à l'utilisateur final.

Des systèmes informatiques embarqués plus évolués ont toutefois été mis en œuvre, ces systèmes étant reprogrammables, comme par exemple les cartes à microprocesseur de type JavaCard. Ces systèmes reprogrammables ajoutent, vis-à-vis des précédents, la possibilité d'enrichir le programme après la mise en service du système, par une opération de téléchargement de fragments de programmes. Ces fragments de programmes, communément désignés par "applets" en langage anglo-saxon, seront indifféremment désignés par appliquettes ou fragments de programmes dans la présente description. Pour une description plus détaillée des systèmes JavaCard, on pourra utilement se reporter à la documentation éditée par la société SUN MICROSYSTEMS INC. Et en particulier à la documentation disponible électroniquement, chapitre JavaCard technology sur le site W.W.W. (World Wide Web) <http://java.sun.com/products/javacard/index.html>, juin 1999.

La figure 1b illustre l'architecture d'un tel système informatique embarqué reprogrammable. Cette architecture est semblable à celle d'un système embarqué classique, à la différence que le système embarqué reprogrammable peut en outre recevoir des appliquettes par l'intermédiaire d'un de ses dispositifs d'entrée/sortie, puis stocker ces dernières dans sa mémoire permanente 13 à partir de laquelle elles peuvent ensuite être exécutées en complément du programme principal.

Pour des raisons de portabilité entre différents systèmes informatiques embarqués, les appliquettes se présentent sous forme de code pour une machine virtuelle standard. Ce code n'est pas directement exécutable par le

microprocesseur 11 mais doit être interprété de manière logicielle par une machine virtuelle 16, laquelle est constituée par un programme résidant en mémoire permanente non inscriptible 12. Dans l'exemple précité des cartes
5 JavaCard, la machine virtuelle utilisée est un sous-ensemble de la machine virtuelle Java. Pour une description des spécifications relatives à la machine virtuelle Java et de la machine virtuelle utilisée, on pourra utilement se reporter à l'ouvrage publié par Tim
10 LINDHOLM et Frank YELLIN, intitulé "*The Java Virtual Machine Specification*", Addison-Wesley 1996, et à la documentation éditée par la société SUN MICROSYSTEMS INC. "*JavaCard 2.1 Virtual Machine Specification*", documentation disponible électroniquement sur le site
15 W.W.W. <http://java.sun.com/products/javacard/JCVMSpec.pdf>, mars 1999.

L'opération de téléchargement d'appliquettes sur un système informatique embarqué en service pose d'importants problèmes de sécurité. Une appliquette
20 involontairement, voire volontairement, mal écrite peut modifier de manière incorrecte des données présentes sur le système, empêcher le programme principal de s'exécuter correctement ou en temps voulu, ou encore modifier d'autres appliquettes téléchargées antérieurement, en
25 rendant celles-ci inutilisables ou nuisibles.

Une appliquette écrite par un pirate informatique peut également divulguer des informations confidentielles stockées dans le système, informations telles que le code d'accès dans le cas d'une carte bancaire par exemple.

A l'heure actuelle, trois solutions ont été proposées en vue de remédier au problème de sécurité des appliquestes.

5 Une première solution consiste à utiliser des signatures cryptographiques afin de n'accepter que des appliquestes provenant de personnes ou d'organismes de confiance.

Dans l'exemple d'une carte bancaire précité, seules les appliquestes portant la signature cryptographique de la
10 banque ayant émis la carte sont acceptées et exécutées par la carte, toute autre appliqueste non signée étant rejetée au cours de l'opération de téléchargement. Un utilisateur mal intentionné de la carte, ne disposant pas de clés de chiffrement de la banque, sera donc dans l'incapacité de
15 faire exécuter une appliqueste non signée et dangereuse sur la carte.

Cette première solution est bien adaptée au cas où toutes les appliquestes proviennent d'une même source unique, la banque dans l'exemple précité. Cette solution est
20 difficilement applicable au cas où les appliquestes proviennent de plusieurs sources, comme, dans l'exemple d'une carte bancaire, le fabricant de la carte, la banque, les organismes gestionnaires des services par carte bancaire, les grands organismes de distribution
25 commerciale offrant à la clientèle des programmes de fidélisation et proposant, légitimement, de télécharger des appliquestes spécifiques sur la carte. Le partage et la détention entre ces divers acteurs économiques des clés de chiffrement nécessaires à la signature électronique des
30 appliquestes posent des problèmes techniques, économiques et juridiques majeurs.

Une deuxième solution consiste à effectuer des contrôles dynamiques d'accès et de typage pendant l'exécution des appliquettes.

Dans cette solution, la machine virtuelle lors de l'exécution des appliquettes, effectue un certain nombre de contrôles, tels que :

- contrôle d'accès à la mémoire : à chaque lecture ou écriture d'une zone mémoire, la machine virtuelle vérifie le droit d'accès de l'appliquette aux données correspondantes ;
- vérification dynamique des types de données : à chaque instruction de l'appliquette, la machine virtuelle vérifie que les contraintes sur les types de données sont satisfaites. A titre d'exemple, la machine virtuelle peut traiter spécialement les données telles que des adresses mémoire valides et empêcher que l'appliquette n'engendre des adresses mémoire invalides par l'intermédiaire de conversions entier/adresse ou d'opérations arithmétiques sur les adresses ;
- détection des débordements de pile et des accès illégaux dans la pile d'exécution de la machine virtuelle, lesquels, dans certaines conditions, sont susceptibles de perturber le fonctionnement de cette dernière, au point de contourner les mécanismes de contrôle précédents.

Cette deuxième solution permet l'exécution d'une large gamme d'appliquettes dans des conditions de sécurité satisfaisantes. Elle présente toutefois l'inconvénient d'un ralentissement considérable de l'exécution, provoqué par l'ensemble des vérifications dynamiques. Pour obtenir une réduction de ce ralentissement, une partie de ces

vérifications peut être prise en charge par le microprocesseur lui-même, au prix toutefois d'une augmentation de la complexité de ce dernier et donc du prix de revient du système embarqué. De telles
5 vérifications augmentent en outre les besoins en mémoire vive et permanente du système, en raison des informations supplémentaires de type qu'il est nécessaire d'associer aux données manipulées.

Une troisième solution consiste à effectuer une
10 vérification statique du code de l'appliquette lors du téléchargement.

Dans cette solution, cette vérification statique simule l'exécution de l'appliquette au niveau des types de données et établit, une fois pour toutes, que le code de
15 l'appliquette respecte la règle des types de données et de contrôle d'accès imposée par la machine virtuelle et ne provoque pas de débordement de pile. Si cette vérification statique réussit, l'appliquette peut alors être exécuté sans qu'il soit nécessaire de vérifier dynamiquement que
20 cette règle est respectée. Dans le cas où le processus de vérification statique échoue, le système embarqué rejette l'"appliquette" et ne permet pas son exécution ultérieure. Pour une description plus détaillée de la troisième solution précitée, on pourra utilement se reporter à
25 l'ouvrage édité par Tim LINDHOLM et Frank YELLIN précédemment cité, à l'article publié par James A.GOSLING intitulé "Java Intermediate Byte Codes", Actes du ACM SIGPLAN, Workshop on Intermediate Representations (IR'95), pages 111-118, janvier 1995, et au brevet US 5,748,964
30 délivré le 05/05/1998.

Vis-à-vis de la deuxième solution, la troisième solution présente l'avantage d'une exécution des appliquestes beaucoup plus rapide, puisque la machine virtuelle n'effectue aucune vérification pendant l'exécution.

La troisième solution présente toutefois l'inconvénient d'un processus de vérification statique du code complexe et coûteux, tant en taille de code nécessaire pour conduire ce processus, qu'en taille de mémoire vive nécessaire pour contenir les résultats intermédiaires de la vérification, et qu'en temps de calcul. A titre d'exemple illustratif, la vérification de code intégré dans le système Java JDK commercialisé par SUN MICROSYSTEMS représente de l'ordre de 50 k-octets de code machine, et sa consommation en mémoire vive est proportionnelle à $(T_p + T_r) \times N_b$ où T_p désigne l'espace maximum de pile, T_r désigne le nombre maximum de registres et N_b désigne le nombre maximum de cibles de branchements utilisées par un sous-programme, encore communément désigné par méthode, de l'appliquette. Ces besoins en mémoire dépassent largement les capacités des ressources de la majorité des systèmes informatiques embarqués actuels, notamment des cartes à microprocesseur commercialement disponibles.

Plusieurs variantes de la troisième solution ont été proposées, dans lesquelles l'éditeur de l'appliquette transmet au vérificateur, outre le code de l'appliquette, un certain nombre d'informations supplémentaires spécifiques telles que types de données précalculés ou preuve préétablie de typage de données correct. Pour une description plus détaillée des modes opératoires

correspondants, on pourra utilement se reporter aux articles publiés par Eva ROSE et Kristoffer HØGSBRO ROSE, "Lightweight Bytecode Verification", Actes du Workshop Formal Underspinning of Java, Octobre 1998, et par George C. NECULA, "Proof-Carrying Code", Actes du 24^e ACM Symposium Principles of Programming Languages, pages 106-119 respectivement.

Ces informations supplémentaires permettent de vérifier le code plus rapidement et de réduire légèrement la taille du code du programme de vérification mais ne permettent toutefois pas de réduire les besoins en mémoire vive, voire les augmentent, de manière très importante, dans le cas des informations de preuve préétablie de typage de données correct.

La présente invention a pour objet de remédier aux inconvénients précités de l'art antérieur.

En particulier, un objet de la présente invention est la mise en œuvre d'un protocole de gestion d'un fragment de programme, ou applique, téléchargé permettant une exécution sur de ce dernier par un système informatique embarqué disposant de faibles ressources, tel qu'une carte à microprocesseur.

Un autre objet de la présente invention est également la mise en œuvre d'un procédé de vérification d'un fragment de programme, ou applique, téléchargé dans lequel un processus de vérification statique du code de l'applique est conduit lors de son téléchargement, ce processus pouvant être rapproché, au moins dans son principe, de la troisième solution précédemment décrite, mais dans lequel des techniques nouvelles de vérification sont introduites, afin de permettre l'exécution de cette

vérification dans les limites de valeurs de taille mémoire et de vitesse de calcul imposées par les cartes à microprocesseur et autres systèmes informatiques embarqués peu puissants.

5 Un autre objet de la présente invention est également la mise en œuvre de procédés de transformation de fragments de programmes de type classique obtenus par exemple par la mise en œuvre d'un compilateur Java en fragments de programmes, ou appliquettes, normalisés,
10 satisfaisant a priori aux critères de vérification du procédé de vérification objet de l'invention, en vue d'accélérer le processus de vérification et d'exécution de ces derniers au niveau des systèmes informatiques embarqués ou cartes à microprocesseur actuels.

15 Un autre objet de la présente invention est, enfin, la réalisation de systèmes informatiques embarqués permettant la mise en œuvre du protocole de gestion et du procédé de vérification d'un fragment de programme téléchargé précités ainsi que de systèmes informatiques
20 permettant la mise en œuvre des procédés de transformation de fragments de programmes, ou appliquettes, classiques en fragments de programmes, ou appliquettes, normalisés précités.

 Le protocole de gestion d'un fragment de programme
25 téléchargé sur un système embarqué reprogrammable, objet de la présente invention, s'applique notamment à une carte à microprocesseur munie d'une mémoire réinscriptible. Le fragment de programme est constitué par un code objet, suite d'instructions, exécutable par le microprocesseur du
30 système embarqué par l'intermédiaire d'une machine virtuelle munie d'une pile d'exécution et de registres ou

variables locales manipulées par ces instructions et permettant d'interpréter ce code objet. Le système embarqué est interconnecté à un terminal.

Il est remarquable en ce qu'il consiste au moins, au
5 niveau du système embarqué, à détecter une commande de téléchargement du fragment de programme. Sur réponse positive à l'étape consistant à détecter une commande de téléchargement, il consiste en outre à lire le code objet constitutif du fragment de programme et à mémoriser
10 temporairement ce code objet dans la mémoire réinscriptible. L'ensemble du code objet mémorisé est soumis à un processus de vérification instruction par instruction. Le processus de vérification consiste au moins en une étape d'initialisation de la pile des types
15 et du tableau des types de registres représentant l'état de la machine virtuelle au début de l'exécution du code objet mémorisé temporairement et en une succession d'étapes de vérification instruction par instruction de l'existence, pour chaque instruction courante, d'une
20 cible, cible d'instruction de branchement, cible d'un gestionnaire d'exceptions, et en une vérification et une actualisation de l'effet de l'instruction courante sur la pile des types et sur le tableau des types de registres. Dans le cas d'une vérification non réussie du code objet,
25 le protocole objet de l'invention consiste à effacer le fragment de programme enregistré momentanément, en l'absence d'enregistrement de ce dernier dans le répertoire de fragments de programmes disponibles, et à adresser au lecteur un code d'erreur.

30 Le procédé de vérification d'un fragment de programme téléchargé sur un système embarqué, objet de

l'invention, s'applique notamment à une carte à microprocesseur munie d'une mémoire réinscriptible. Le fragment de programme est constitué par un code objet et comporte au moins un sous-programme, suite d'instructions, exécutable par le microprocesseur du système embarqué par l'intermédiaire d'une machine virtuelle munie d'une pile d'exécution et de registres d'opérandes manipulés par ces instructions et permettant d'interpréter ce code objet. Le système embarqué est interconnecté à un lecteur.

Il est remarquable en ce que, suite à la détection d'une commande de téléchargement et à la mémorisation du code objet constitutif du fragment de programme dans la mémoire réinscriptible, il consiste, pour chaque sous-programme, à effectuer une étape d'initialisation de la pile des types et du tableau des types de registres par des données représentant l'état de la machine virtuelle au début de l'exécution du code objet mémorisé temporairement, à effectuer une vérification du code objet mémorisé temporairement instruction par instruction, par discrimination de l'existence, pour chaque instruction courante, d'une cible d'instruction de branchement, d'une cible d'un appel d'un gestionnaire d'exceptions ou d'une cible d'un appel de sous-routine et à effectuer une vérification et une actualisation de l'effet de l'instruction courante sur les types de données de la pile des types et du tableau des types de registres, en fonction de l'existence d'une cible d'instruction de branchement, d'une cible d'un appel de sous-routine ou d'une cible d'un appel de gestionnaire d'exceptions. La vérification est réussie lorsque le tableau des types de registres n'est pas modifié au cours d'une vérification de

toutes les instructions, le processus de vérification étant poursuivi instruction par instruction jusqu'à ce que le tableau des types de registres soit stable, en l'absence de modification. Le processus de vérification est interrompu sinon.

Le procédé de transformation d'un code objet d'un fragment de programme en un code objet normalisé pour ce même fragment de programme, objet de la présente invention, s'applique à un code objet d'un fragment de programme dans lequel les opérandes de chaque instruction appartiennent aux types de données manipulées par cette instruction, la pile d'exécution ne présente pas de phénomène de débordement et pour chaque instruction de branchement le type des variables de la pile au niveau de ce branchement est le même qu'au niveau des cibles de ce branchement. Le code objet normalisé obtenu est tel que les opérandes de chaque instruction appartiennent aux types de données manipulées par cette instruction, la pile d'exécution ne présente pas de phénomène de débordement et la pile d'exécution est vide à chaque instruction de cible de branchement.

Il est remarquable en ce qu'il consiste, pour l'ensemble des instructions du code objet, à annoter chaque instruction courante par le type de données de la pile d'exécution avant et après l'exécution de cette instruction, les données d'annotation étant calculées au moyen d'une analyse du flot des données relatif à cette instruction, à détecter au sein des instructions, et de chaque instruction courante, l'existence de branchements pour lesquels la pile d'exécution n'est pas vide, l'opération de détection étant conduite à partir des

données d'annotation du type de variables de pile allouées à chaque instruction courante. En présence d'une détection d'une pile d'exécution non vide, il consiste en outre à insérer des instructions de transfert des variables de pile de part et d'autre de ces branchements ou de ces cibles de branchement afin de vider le contenu de la pile d'exécution dans des registres temporaires avant ce branchement et de rétablir la pile d'exécution à partir des registres temporaires après ce branchement et à n'insérer aucune instruction de transfert sinon.

Ce procédé permet ainsi d'obtenir un code objet normalisé pour ce même fragment de programme, dans lequel la pile d'exécution est vide à chaque instruction de branchement et de cible de branchement, en l'absence de toute modification de l'exécution du fragment de programme.

Le procédé de transformation d'un code objet d'un fragment de programme en un code objet normalisé pour ce même fragment de programme, objet de la présente invention, s'applique, en outre, à un code objet d'un fragment de programme dans lequel les opérandes de chaque instruction appartiennent aux types de données manipulées par cette instruction, et un opérande de type déterminé écrit dans un registre par une instruction de ce code objet est relu depuis ce même registre par une autre instruction de ce code objet avec le même type de donnée déterminé. Le code objet normalisé obtenu est tel que les opérandes appartiennent aux types de données manipulées par cette instruction, un seul et même type de donnée étant alloué à un même registre dans tout le code objet normalisé.

Il est remarquable en ce qu'il consiste, pour l'ensemble des instructions du code objet, à annoter chaque instruction courante par le type de donnée des registres avant et après l'exécution de cette instruction, les données d'annotation étant calculées au moyen d'une analyse du flot des données relatif à cette instruction, et à effectuer une réallocation des registres d'origine employés avec des types différents, par division de ces registres d'origine en registres normalisés distincts. Un registre normalisé est alloué à chaque type de donnée utilisé. Une réactualisation des instructions qui manipulent les opérandes qui font appel aux registres normalisés est effectuée.

Le protocole de gestion d'un fragment de programme, le procédé de vérification d'un fragment de programme, les procédés de transformation de code objet de fragments de programmes en code objet normalisé et les systèmes correspondants, objets de la présente invention, trouvent application au développement des systèmes embarqués reprogrammables, tels que les cartes à microprocesseur, notamment dans l'environnement Java.

Il seront mieux compris à la lecture de la description et à l'observation des dessins ci-après, dans lesquels, outre les figures 1a et 1b relatives à l'art antérieur :

- la figure 2 représente un organigramme illustratif du protocole de gestion d'un fragment de programme téléchargé sur un système embarqué reprogrammable,
- la figure 3a représente, à titre illustratif, un organigramme d'un procédé de vérification d'un fragment

de programme téléchargé conformément à l'objet de la présente invention,

- 5 - la figure 3b représente un diagramme illustratif des types de données et des relations de sous-typage mis en œuvre par le procédé de gestion et le procédé de vérification d'un fragment de programme téléchargé, objet de la présente invention,
- 10 - la figure 3c représente un détail du procédé de vérification selon la figure 3a, relatif à la gestion d'une instruction de branchement,
- la figure 3d représente un détail du procédé de vérification selon la figure 3a, relatif à la gestion d'une instruction d'appel de sous-routine,
- 15 - la figure 3e représente un détail du procédé de vérification selon la figure 3a, relatif à la gestion d'une cible d'un gestionnaire d'exceptions,
- la figure 3f représente un détail du procédé de vérification selon la figure 3a, relatif à la gestion d'une cible de branchements incompatibles,
- 20 - la figure 3g représente un détail du procédé de vérification selon la figure 3a, relatif à la gestion d'une absence de cible de branchement,
- la figure 3h représente un détail du procédé de vérification selon la figure 3a, relatif à la gestion de l'effet de l'instruction courante sur la pile des types,
- 25 - la figure 3i représente un détail du procédé de vérification selon la figure 3a, relatif à la gestion d'une instruction de lecture d'un registre,

- la figure 3j représente un détail du procédé de vérification selon la figure 3a, relatif à la gestion d'une instruction d'écriture d'un registre,
- 5 - la figure 4a représente un organigramme illustratif d'un procédé de transformation d'un code objet d'un fragment de programme en un code objet normalisé pour ce même fragment de programme à instruction de branchement respectivement de cible de branchement à pile vide,
- 10 - la figure 4b représente un organigramme illustratif d'un procédé de transformation d'un code objet d'un fragment de programme en un code objet normalisé pour ce même fragment de programme faisant appel à des registres typés, à chaque registre étant attribué un
- 15 - seul type de donnée spécifique,
- la figure 5a représente un détail de mise en œuvre du procédé de transformation illustré en figure 4a,
- la figure 5b représente un détail de mise en œuvre du procédé de transformation illustré en figure 4b,
- 20 - la figure 6 représente un schéma fonctionnel de l'architecture complète d'un système de développement d'un fragment de programme normalisé, et d'une carte à microprocesseur reprogrammable permettant la mise en œuvre du protocole de gestion et du procédé de
- 25 - vérification d'un fragment de programme conformément à l'objet de la présente invention.

D'une manière générale, on indique que le protocole de gestion, le procédé de vérification et de transformation d'un fragment de programme téléchargé, 30 objet de la présente invention, ainsi que les systèmes correspondants, sont mis en œuvre grâce à une architecture

logicielle pour le téléchargement et l'exécution sûrs d'appliquettes sur un système informatique embarqué disposant de faibles ressources, tel que notamment les cartes à microprocesseur.

5 D'une manière générale, on indique que la description ci-après concerne l'application de l'invention dans le contexte des cartes à microprocesseur reprogrammables de type JavaCard, confer documentation disponible électroniquement auprès de la société SUN
10 MICROSYSTEMS INC. rubrique JavaCard Technology précédemment mentionnée dans la description.

Toutefois, la présente invention est applicable à tout système embarqué reprogrammable par l'intermédiaire d'un téléchargement d'appliquette écrite dans le code
15 d'une machine virtuelle comprenant une pile d'exécution, des registres ou variables locales et dont le modèle d'exécution est fortement typé, chaque instruction du code de l'appliquette ne s'appliquant qu'à des types de données spécifiques. Le protocole de gestion d'un fragment de
20 programme téléchargé sur un système embarqué reprogrammable, objet de la présente invention, sera maintenant décrit de manière plus détaillée en liaison avec la figure 2.

En liaison avec la figure précitée, on indique que
25 le code objet constitutif du fragment de programme ou appliquette est constitué par une suite d'instructions exécutables par le microprocesseur du système embarqué par l'intermédiaire de la machine virtuelle précédemment mentionnée. La machine virtuelle permet d'interpréter le
30 code objet précité. Le système embarqué est interconnecté

à un terminal par l'intermédiaire d'une liaison série par exemple.

En référence à la figure 2 précédemment mentionnée, le protocole de gestion objet de la présente invention consiste au moins, au niveau du système embarqué, à détecter en une étape 100a, 100b, une commande de téléchargement de ce fragment de programme. Ainsi, l'étape 100a peut consister en une étape de lecture de la commande précitée et l'étape 100b en une étape de test de la commande lue et de vérification de l'existence d'une commande de téléchargement.

Sur réponse positive à l'étape de détection 100a, 100b précitée d'une commande de téléchargement, le protocole objet de la présente invention consiste ensuite à lire à l'étape 101 le code objet constitutif du fragment de programme considéré et à mémoriser temporairement le code objet précité dans la mémoire du système informatique embarqué. L'opération de mémorisation temporaire précitée peut être exécutée soit dans la mémoire réinscriptible, soit, le cas échéant, dans la mémoire vive du système embarqué, lorsque cette dernière présente une capacité suffisante. L'étape de lecture du code objet et de mémorisation temporaire de ce dernier dans la mémoire réinscriptible est désignée par chargement du code de l'appliquette sur la figure 2.

L'étape précitée est alors suivie d'une étape 102 consistant à soumettre l'ensemble du code objet mémorisé temporairement à un processus de vérification, instruction par instruction, du code objet précité.

Le processus de vérification consiste au moins en une étape d'initialisation de la pile des types et du

tableau des types de données représentant l'état de la machine virtuelle au début de l'exécution du code objet mémorisé temporairement, ainsi qu'en une succession d'étapes de vérification instruction par instruction par discrimination de l'existence, pour chaque instruction courante, notée I_i , d'une cible telle qu'une cible d'instruction de branchement notée CIB, une cible d'appel d'un gestionnaire d'exceptions ou une cible d'un appel de sous-routine. Une vérification et une actualisation de l'effet de l'instruction courante I_i sur la pile des types et sur le tableau des types de registres est effectuée.

Lorsque la vérification est réussie à l'étape 103a, le protocole objet de la présente invention consiste à enregistrer à l'étape 104 le fragment de programme téléchargé dans un répertoire de fragments de programmes disponibles et à envoyer à l'étape 105 au lecteur un accusé de réception positif.

Au contraire, dans le cas d'une vérification non réussie du code objet à l'étape 103b, le protocole objet de l'invention consiste à inhiber, en une étape 103c, toute exécution, sur le système embarqué du fragment de programme enregistré momentanément. L'étape d'inhibition 103c peut être mise en œuvre de différentes manières. Cette étape peut, à titre d'exemple non limitatif, consister à effacer à l'étape 106 le fragment de programme enregistré momentanément en l'absence d'enregistrement de ce fragment de programme dans le répertoire de fragments de programmes disponibles et, à une étape 107, à adresser au lecteur un code d'erreur. Les étapes 107 et 105 peuvent être réalisées soit séquentiellement après les

étapes 106 respectivement 104, soit en opération multitâche avec celles-ci.

En référence à la même figure 2, sur réponse négative à l'étape consistant à détecter une commande de téléchargement à l'étape 100b, le protocole objet de la présente invention consiste à détecter, en une étape 108, une commande de sélection d'un fragment de programme disponible dans un répertoire de fragments de programmes et, sur réponse positive à l'étape 108, la sélection d'un fragment de programme disponible étant détectée, à appeler à l'étape 109 ce fragment de programme disponible sélectionné en vue de son exécution. L'étape 109 est alors suivie d'une étape d'exécution 110 du fragment de programme disponible appelé par l'intermédiaire de la machine virtuelle en l'absence de toute vérification dynamique de types de variables, des droits d'accès aux objets manipulés par le fragment de programme disponible appelé ou du débordement de la pile d'exécution lors de l'exécution de chaque instruction.

Dans le cas où une réponse négative est obtenue à l'étape 108, cette étape consistant à détecter une commande de sélection d'un fragment de programme disponible appelé, le protocole objet de la présente invention consiste à procéder, à une étape 111, au traitement des commandes standard du système embarqué.

En ce qui concerne l'absence de vérification dynamique de type ou de droit d'accès aux objets de type JavaCard par exemple, on indique que cette absence de vérification ne compromet pas la sûreté de la carte car le code de l'appliquette a nécessairement passé avec succès la vérification.

D'une manière plus spécifique, on indique que la vérification de code effectuée, conformément au procédé objet de l'invention, sur la carte à microprocesseur ou sur le système informatique embarqué est plus sélective
5 que la vérification habituelle de codes pour la machine virtuelle Java telle que décrite dans l'ouvrage intitulé "The Java Virtual Machine Specification" précédemment mentionné dans la description.

Toutefois, tout code de la machine virtuelle Java
10 correct au sens du vérificateur Java traditionnel peut être transformé en un code équivalent susceptible de passer avec succès la vérification de code effectuée sur la carte à microprocesseur.

Alors qu'il est possible d'envisager l'écriture
15 directe de codes Java satisfaisant aux critères de vérification précédemment mentionnés dans le cadre de la mise en œuvre du protocole objet de la présente invention, un objet remarquable de celle-ci est également la mise en œuvre d'un procédé de transformation automatique de tout
20 code Java standard en un code normalisé pour le même fragment de programme satisfaisant nécessairement aux critères de vérification mis en œuvre précédemment cités. Le procédé de transformation en code normalisé et le système correspondant seront décrits ultérieurement dans
25 la description de manière détaillée.

Une description plus détaillée du procédé de vérification d'un fragment de programme, ou appliquette, conforme à l'objet de la présente invention, sera maintenant donnée en liaison avec la figure 3a et les
30 figures suivantes.

D'une manière générale, on indique que le procédé de vérification objet de la présente invention peut être mis en œuvre soit dans le cadre du protocole de gestion d'un fragment de programme objet de l'invention
5 précédemment décrit en liaison avec la figure 2, soit de manière indépendante, afin d'assurer tout processus de vérification jugé nécessaire.

D'une manière générale, on indique qu'un fragment de programme est constitué par un code objet comportant au moins un sous-programme, plus communément désigné par
10 méthode, et constitué par une suite d'instructions exécutables par le microprocesseur du système embarqué par l'intermédiaire de la machine virtuelle.

Ainsi que représenté sur la figure 3a, le procédé
15 de vérification consiste, pour chaque sous-programme, à effectuer une étape 200 d'initialisation de la pile des types et du tableau des types de registres de la machine virtuelle par des données représentant l'état de cette machine virtuelle au début de l'exécution du code objet,
20 objet de la vérification. Ce code objet peut être mémorisé temporairement ainsi que décrit précédemment en liaison avec la mise en œuvre du protocole objet de la présente invention.

L'étape 200 précitée est alors suivie d'une étape
25 200a consistant à positionner la lecture de l'instruction courante I_i , index i , sur la première instruction du code objet. L'étape 200a est suivie d'une étape 201 consistant à effectuer une vérification du code objet précité instruction par instruction, par discrimination pour
30 chaque instruction courante notée I_i de l'existence d'une cible d'instruction de branchement CIB, d'une cible d'un

appel de gestionnaire d'exceptions, noté CEM, ou d'une cible d'un appel de sous-routine CSR.

L'étape de vérification 201 est suivie d'une étape de vérification 202 et d'actualisation de l'effet de l'instruction courante I_i sur les types de données de la pile des types et du tableau des types de registres en fonction de l'existence, pour l'instruction courante visée par une autre instruction, d'une cible d'instruction de branchement CIB, d'une cible d'un appel de sous-routine CSR ou d'une cible d'un appel de gestionnaire d'exceptions CEM.

L'étape 202 pour l'instruction courante I_i est suivie d'une étape de test 203 d'atteinte de la dernière instruction, test noté :

I_i = dernière instruction du code objet ?
Sur réponse négative au test 203, le processus passe à l'instruction suivante 204, noté $i = i+1$, et au retour à l'étape 201.

On indique que la vérification précitée, à l'étape 202, est réussie lorsque le tableau des types de registres n'est pas modifié au cours d'une vérification de toutes les instructions I_i constitutives du code objet. Dans ce but, un test 205 d'existence d'un état stable du tableau des types de registres et prévu. Ce test est noté :

$\exists ?$ Etat stable du tableau des types de registres.
Sur réponse positive au test 205, la vérification a réussi.

Dans le cas où au contraire aucune absence de modification est constatée, le processus de vérification est réitéré et relancé par retour à l'étape 200a. On démontre que la fin du processus est garantie après au

plus $NrxH$ itérations, où Nr désigne le nombre de registres utilisés et H une constante dépendant de la relation de sous-typage.

5 Différentes indications relatives aux types de variables manipulées au cours du processus de vérification décrit précédemment en liaison avec la figure 3a seront maintenant données en liaison avec la figure 3b.

10 Les types de variables précités comportent au moins des identificateurs de classes correspondant aux classes d'objets définis dans le fragment de programme soumis à la vérification, des types de variables numériques comportant au moins un type short, entier codé sur p bits, p pouvant prendre la valeur $p = 16$, et un type d'adresse de retour d'une instruction de saut JSR, ce type
15 d'adresse étant noté retaddr, un type null relatif à des références d'objets nuls, un type object relatif aux objets proprement dits, un type spécifique \perp représentant l'intersection de tous les types et correspondant à la valeur zéro nul, un autre type spécifique \top représentant
20 l'union de tous les types et correspondant à tout type de valeurs.

En référence à la figure 3b, on indique que l'ensemble des types de variables précités vérifient une relation de sous-typage :

25 object $\in T$;
short, retaddr $\in T$;
 \perp \in null, short, retaddr

30 Un exemple plus spécifique d'un processus de vérification tel qu'illustré en figure 3a sera maintenant donné en liaison avec un premier exemple de structure de données illustré au tableau T1 joint en annexe.

L'exemple précité concerne une appliquette écrite en code Java.

Le processus de vérification accède au code du sous-programme constitutif de l'appliquette soumis à
5 vérification par l'intermédiaire d'un pointeur sur l'instruction I_i en cours de vérification.

Le processus de vérification enregistre la taille et le type de la pile d'exécution au niveau de l'instruction courante I_i correspondant à saload sur
10 l'exemple du tableau T1 précité.

Le processus de vérification enregistre alors la taille et le type de la pile d'exécution au niveau de l'instruction courante dans la pile des types par l'intermédiaire de son pointeur de pile des types.

15 Ainsi que mentionné précédemment dans la description, cette pile de types reflète l'état de la pile d'exécution de la machine virtuelle au niveau de l'instruction courante I_i . Dans l'exemple représenté au tableau T1, lors de l'exécution future de l'instruction
20 I_i , la pile contiendra trois entrées : une référence vers un objet de la classe C, une référence vers un tableau d'entiers codés sur $p = 16$ bits, le type short[], et un entier de $p = 16$ bits de type short. Ceci est également représenté dans la pile des types qui contient également
25 trois entrées : C, le type des objets de la classe C, short[], le type des tableaux d'entiers $p = 16$ bits et short, le type des entiers $p = 16$ bits.

Une autre structure de données remarquable est constituée par un tableau de types de registres, ce
30 tableau reflétant l'état des registres, c'est-à-dire des

registres mémorisant les variables locales, de la machine virtuelle.

En continuant l'exemple indiqué au tableau T1, on indique que l'entrée 0 du tableau de types de registres contient le type C, c'est-à-dire que lors de l'exécution future de l'instruction courante $I_i = \text{saload}$, le registre 0 est assuré de contenir une référence vers un objet de la classe C.

Les différents types manipulés au cours de la vérification et stockés dans le tableau de types de registres et dans la pile de types sont représentés en figure 3b. Ces types comprennent :

- des identificateurs de classe CB correspondant aux classes d'objets spécifiques définis dans l'appliquette ;
- des types de base, tels que short entier codé sur $p = 16$ bits, int1 et int2, p bits les plus et les moins significatifs respectivement d'entiers codés sur $2p$ bits par exemple, ou retaddr adresse de retour d'une instruction, ainsi que mentionné précédemment ;
- le type null représentant les références d'objets nuls.

En ce qui concerne la relation de sous-typage, on indique qu'un type T1 est sous-type d'un type T2 si toute valeur valide du type T1 est également une valeur valide du type T2. Le sous-typage entre identificateur de classes reflète la hiérarchie d'héritage entre classes de l'appliquette. Sur les autres types, le sous-typage est défini par le treillis représenté en figure 3b, \perp étant sous-type de tous les types et tous les types étant sous-types de \top .

Le déroulement du processus de vérification d'un sous-programme constitutif d'une appliquette est le suivant, en référence au tableau T1 précédemment mentionné.

5 Le processus de vérification s'effectue indépendamment sur chaque sous-programme de l'appliquette. Pour chaque sous-programme, le processus effectue une ou plusieurs passes de vérification sur les instructions du sous-programme considéré. Le pseudo-code du processus de
10 vérification est donné au tableau T2 joint en annexe.

 Le processus de vérification d'un sous-programme débute par l'initialisation de la pile des types et du tableau des types de registres représentés au tableau T1, cette initialisation reflétant l'état de la machine
15 virtuelle au début de l'exécution du sous-programme examiné.

 La pile des types est initialement vide, le pointeur de pile est égal à zéro, et les types de registres sont initialisés avec les types des paramètres
20 du sous-programme illustrant le fait que la machine virtuelle passe les paramètres de ce sous-programme dans ces registres. Les types de registres alloués par le sous-programme sont initialisés aux types de données 1 illustrant le fait que la machine virtuelle initialise ces
25 registres à zéro au début de l'exécution du sous-programme.

 Ensuite, une ou plusieurs passes de vérification sur les instructions et sur chaque instruction courante I_i du sous-programme sont effectuées.

30 A la fin de la passe de vérification mise en œuvre ou d'une succession de passes par exemple, le processus de

vérification détermine si les types de registres contenus dans le tableau des types de registres représentés au tableau T1 de l'annexe ont changé pendant la passe de vérification. En l'absence de changement, la vérification
5 est terminée et un code de succès est renvoyé au programme principal, lequel permet d'envoyer l'accusé de réception positif à l'étape 105 du protocole de gestion représenté en figure 2.

En présence d'une modification du tableau des
10 types de registres précité, le processus de vérification réitère la passe de vérification jusqu'à ce que les types de registres contenus dans le tableau des types de registres soit stable.

Le déroulement proprement dit d'une passe de
15 vérification effectuée une ou plusieurs fois jusqu'à la stabilité du tableau des types de registres sera maintenant décrit en liaison avec les figures 3c à 3j.

Pour chaque instruction courante I_i , les vérifications suivantes sont effectuées :

20 En liaison avec la figure 3a à l'étape 201, le processus de vérification détermine si l'instruction courante I_i est la cible d'une instruction de branchement, d'appel de sous-routine ou d'un gestionnaire d'exception, ainsi que mentionné précédemment. Cette vérification
25 s'effectue par examen des instructions de branchement contenues dans le code du sous-programme et des gestionnaires d'exceptions associés à ce sous-programme.

En référence à la figure 3c ouverte par l'étape
30 201, lorsque l'instruction courante I_i est la cible d'une instruction de branchement, cette condition étant réalisée par un test 300 désigné par $I_i = \text{CIB}$, ce branchement étant

inconditionnel ou conditionnel, le processus de vérification s'assure que la pile des types est vide à ce point du sous-programme par un test 301. Sur réponse positive au test 301, le processus de vérification est poursuivi par une étape suite de contexte noté suite A. Sur réponse négative au test 301, la pile des types n'étant pas vide, la vérification échoue et l'appliquette est rejetée. Cet échec est représentée par l'étape Echec.

En référence à la figure 3d ouverte par l'étape suite A, lorsque l'instruction courante I_i est la cible d'un appel de sous-routine, cette condition étant réalisée par un test 304 $I_i = \text{CSR}$, le processus de vérification vérifie en un test 305 que l'instruction précédente I_{i-1} ne continue pas en séquence. Cette vérification est réalisée par une étape de test 305 lorsque l'instruction précédente constitue un branchement inconditionnel, un retour de sous-routine ou une levée d'exception. Le test à l'étape 305 est noté ainsi :

$I_{i-1} = \text{IB}_{\text{inconditionnel}}$, retour RSR ou levée L-EXCEPT.

Sur réponse négative au test 305, le processus de vérification échoue en une étape Echec. Au contraire, sur réponse positive au test 305, le processus de vérification réinitialise la pile des types de façon que celle-ci contienne exactement une entrée de type retaddr, adresse de retour de la sous-routine précitée. Si l'instruction courante I_i à l'étape 304 n'est pas la cible d'un appel de sous-routine, le processus de vérification est poursuivi dans le contexte à l'étape suite B.

En référence à la figure 3e, lorsque l'instruction courante I_i est la cible d'un gestionnaire d'exceptions, cette condition étant réalisée par un test 307 noté $I_i =$

CEM, CEM désignant la cible d'un gestionnaire d'exceptions, cette condition est réalisée par l'intermédiaire d'un test 307, noté :

$$I_i = \text{CEM}.$$

- 5 Le processus, sur réponse positive au test 307 vérifie que l'instruction précédente constitue un branchement inconditionnel, un retour de sous-routine ou une levée d'exceptions par l'intermédiaire d'un test 305 noté :

$$I_{i-1} = \text{IB}_{\text{inconditionnel}}, \text{ retour RSR ou levée L-EXCEPT.}$$

- 10 Sur réponse positive au test 305, le processus de vérification procède à une réactualisation de la pile des types, à une étape 308, par une entrée de types des exceptions, notée type EXCEPT, l'étape 308 étant suivie d'une étape de suite de contexte, suite C. Sur réponse
15 négative au test 305, le processus de vérification échoue par l'étape notée Echec. Le fragment de programme est alors rejeté.

- En référence à la figure 3f, lorsque l'instruction courante I_i est la cible d'une pluralité de branchements
20 incompatibles, cette condition est réalisée par un test 309, lequel est noté :

$$I_i = \text{XIB incompatibles}$$

- les branchements incompatibles étant par exemple un branchement inconditionnel et un appel de sous-routine ou
25 encore deux gestionnaires d'exceptions différents. Sur réponse positive au test 309, les branchements étant incompatibles, le processus de vérification échoue par une étape notée Echec et le fragment de programme est rejeté. Sur réponse négative au test 309, le processus de
30 vérification est poursuivi par une étape notée suite D. Le

test 309 est ouvert par l'étape suite C précédemment mentionnée dans la description.

En référence à la figure 3g, lorsque l'instruction courante I_i n'est la cible d'aucun branchement, cette condition étant réalisée par un test 310 ouvert par la suite D précédemment mentionnée, ce test étant noté

$I_i \exists ?$ cibles de branchement,

\exists désignant le symbole d'existence,

le processus de vérification continue sur réponse négative au test 310 par passage à une actualisation de la pile des types à une étape 311, l'étape 311 et la réponse positive au test 310 étant suivies d'une étape de suite de contexte à l'étape 202, décrite précédemment dans la description en liaison avec la figure 3a.

Une description plus détaillée de l'étape de vérification de l'effet de l'instruction courante sur la pile des types à l'étape 202 précédemment citée sera maintenant donnée en liaison avec la figure 3h.

Selon la figure précitée, cette étape peut comporter au moins une étape 400 de vérification que la pile d'exécution des types contient au moins autant d'entrées que l'instruction courante comporte d'opérandes. Cette étape de test 400 est notée :

$$Nbep \geq NOpi$$

où Nbep désigne le nombre d'entrées de la pile des types et NOpi désigne le nombre d'opérandes contenus dans l'instruction courante.

Sur réponse positive au test 400, ce test est suivi d'une étape 401a de dépilement de la pile des types et de vérification 401b que les types des entrées au sommet de la pile sont sous-types des types des opérandes de

l'instruction courante précitée. A l'étape de test 401a, les types des opérandes de l'instruction i sont notés $TOpi$ et les types des entrées au sommet de la pile sont notés Targs.

5 A l'étape 401b, la vérification correspond à une vérification de la relation de sous-typage Targs sous-type de $TOpi$.

 Sur réponse négative au test 400 et au test 401b, le processus de vérification échoue, ce qui est illustré
10 par l'accès à l'étape Echec. Au contraire sur réponse positive au test 401b, le processus de vérification est poursuivi et consiste à effectuer :

 - Une étape de vérification de l'existence d'un espace mémoire suffisant sur la pile des types, pour
15 procéder à l'empilement des résultats de l'instruction courante. Cette étape de vérification est réalisée par un test 402 noté :

$Esp\text{-}pile \geq Esp\text{-}résultats$

20 où chaque membre de l'inégalité désigne l'espace mémoire correspondant.

Sur réponse négative au test 402, le processus de vérification échoue, ce qui est illustré par l'étape Echec. Au contraire, sur réponse positive au test 402, le processus de vérification procède alors à l'empilement des
25 types de données attribuées aux résultats en une étape 403, l'empilement étant effectué sur la pile des types de données attribuée à ces résultats.

 A titre d'exemple non limitatif, on indique que pour la mise en œuvre de la figure 3h de vérification de
30 l'effet de l'instruction courante sur la pile des types, pour une instruction courante constituée par une

instruction Java saload correspondant à la lecture d'un élément entier codé sur $p = 16$ bits dans un tableau d'entiers, ce tableau d'entiers étant défini par le tableau d'entiers et un indice entier dans ce tableau, et
5 le résultat par l'entier lu à cet indice dans ce tableau, le processus de vérification s'assure que la pile des types contient au moins deux éléments, que les deux éléments au sommet de la pile des types sont sous-types de short[] respectivement short, procède au processus de
10 dépilement et ensuite au processus d'empilement du type de données short comme type du résultat.

En outre, en référence à la figure 3i, pour la mise en œuvre de l'étape de vérification de l'effet de l'instruction courante sur la pile des types, lorsque
15 l'instruction courante I_i est une instruction de lecture, notée IR, d'un registre d'adresse n , cette condition étant réalisée par un test 404 noté $I_i = IR_n$, sur réponse positive au test 404 précité, le processus de vérification consiste à vérifier le type de données du résultat de cette
20 lecture, en une étape 405, par consultation de l'entrée n du tableau des types de registres, puis à déterminer l'effet de l'instruction courante I_i sur la pile des types par une opération 406a de dépilement des entrées de la pile correspondant aux opérandes de cette instruction
25 courante et par empilement 406b du type de données de ce résultat. Les opérandes de l'instruction I_i sont notés OP_i . Les étapes 406a et 406b sont suivies d'un retour à la suite de contexte suite F. Sur réponse négative au test 404, le processus de vérification est poursuivi par la
30 suite de contexte suite F.

En référence à la figure 3j, lorsque l'instruction courante I_i est une instruction d'écriture, notée IW, d'un registre d'adresse n, cette condition étant réalisée par un test noté $I_i = IW_m$, le processus de vérification
5 consiste, sur réponse positive au test 407, à déterminer en une étape 408 l'effet de l'instruction courante sur la pile des types et le type t de l'opérande écrit dans le registre d'adresse n, puis, en une étape 409, à remplacer l'entrée du type du tableau des types de registres à
10 l'adresse n par le type immédiatement supérieur au type précédemment stocké et au type t de l'opérande écrit dans le registre d'adresse n. L'étape 409 est suivie d'un retour à la suite de contexte suite 204. Sur réponse négative au test 407, le processus de vérification est
15 poursuivi par une suite de contexte suite 204.

A titre d'exemple, lorsque l'instruction courante I_i correspond à l'écriture d'une valeur de type D dans un registre d'adresse 1 et que le type du registre 1 avant
vérification de l'instruction était C, le type du registre
20 1 est remplacé par le type object qui est le type supérieur le plus petit de C et D dans le treillis des types représenté en figure 3b.

De même, à titre d'exemple, lorsque l'instruction courante I_i est une lecture d'une instruction aload-0
25 consistant à empiler le contenu du registre 0 et que l'entrée 0 du tableau des types de registres est C, le vérificateur empile C sur la pile des types.

Un exemple de vérification d'un sous-programme écrit en environnement Java sera maintenant donné en
30 liaison avec les tableaux T3 et T4 introduits en annexe.

Le tableau T3 représente un code JavaCard spécifique correspondant au sous-programme Java inclus dans ce tableau.

Le tableau T4 illustre le contenu du tableau des types de registres et de la pile des types avant la vérification de chaque instruction. Les contraintes de types sur les opérandes des diverses instructions sont toutes respectées. La pile est vide aussi bien après l'instruction de branchement 5 à l'instruction 9 symbolisée par la flèche, qu'avant la cible de branchement 9 précitée. Le type du registre 1 qui était initialement 1 devient null, la borne supérieure de null et de 1, lorsque l'instruction 1 de stockage d'une valeur de type null dans le registre 1 est examinée, puis devient de type short[], la borne supérieure du type short[] et du type null, lorsque l'instruction 8, stockage d'une valeur de type short[] dans le registre 1 est traitée. Le type du registre 1 ayant changé pendant la première passe de vérification, une seconde passe est effectuée en repartant des types de registres obtenus à la fin de la première. Cette seconde passe de vérification réussit tout comme la première et ne modifie pas les types des registres. Le processus de vérification se termine donc avec succès.

Différents exemples de cas d'échec du processus de vérification sur quatre exemples de code incorrect seront maintenant donnés en liaison avec le tableau T5 introduit en annexe :

- Au point a) du tableau T5, le code donné en exemple a pour objet de tenter de fabriquer une référence d'objet invalide en utilisant un processus arithmétique sur des pointeurs. Il est rejeté par la vérification des types

des arguments de l'instruction 2 sadd, laquelle exige que ces deux arguments soient de type short.

- Aux points b) et c) du tableau T5, le code a pour objet de réaliser deux tentatives de convertir un entier quelconque en une référence d'objet. Au point b), le registre 0 est utilisé à la fois avec le type short, instruction 0, et avec le type null, instruction 5. En conséquence, le processus de vérification attribue le type I au registre 0 et détecte une erreur de type I lorsque le registre 0 est renvoyé comme résultat de type object à l'instruction 7.
- Au point c) du tableau T5, un ensemble de branchements du type "if...then...else..." est utilisé pour laisser au sommet de la pile un résultat qui est constitué soit par un entier soit par une référence d'objet. Le processus de vérification rejette ce code car il détecte que la pile n'est pas vide au niveau du branchement de l'instruction 5 vers l'instruction 9 symbolisée par la flèche.
- Enfin, au point d) du tableau T5, le code contient une boucle qui, à chaque itération, a pour effet d'empiler un entier de plus au sommet de la pile et de provoquer donc un débordement de la pile après un certain nombre d'itérations. Le processus de vérification rejette ce code en constatant que la pile n'est pas vide au niveau du branchement arrière de l'instruction 8 vers l'instruction 0, symbolisé par la flèche de retour, la pile n'étant pas vide à un point de branchement.

Les différents exemples donnés précédemment en liaison avec les tableaux T3, T4 et T5 montrent que le processus de vérification, objet de la présente invention,

est particulièrement efficace et qu'il s'applique à des appliquettes et en particulier aux sous-programmes de ces dernières, pour lesquelles les conditions de type de pile respectivement de caractère vide de la pile des types antérieurement et aux instructions de branchement ou de cibles de branchement sont satisfaites.

Bien entendu, un tel processus de vérification implique l'écriture de codes objet satisfaisant à ces critères, ces codes objet pouvant correspondre au sous-programme introduit au tableau T3 précédemment mentionné.

Toutefois, et afin d'assurer la vérification d'appliquettes et de sous-programmes d'appliquettes existantes qui ne satisfont pas nécessairement aux critères de vérification du procédé objet de la présente invention, en particulier pour ce qui concerne les appliquettes et les sous-programmes écrits en environnement Java, la présente invention a pour objet d'établir des procédés de transformation de ces appliquettes ou sous-programmes en appliquettes ou fragments de programme normalisés permettant de subir avec succès les tests de vérification du procédé de vérification objet de la présente invention et du protocole de gestion mettant en œuvre un tel procédé.

Dans ce but, l'invention a donc pour objet la mise en œuvre d'un procédé et d'un programme de transformation d'un code objet classique constituant une appliquette, ce procédé et ce programme de transformation pouvant être mis en œuvre hors d'un système embarqué ou d'une carte à microprocesseur lors de la création de l'appliquette considérée.

Le procédé de transformation de code en code normalisé objet de la présente invention, sera maintenant décrit dans le cadre de l'environnement Java à titre d'exemple purement illustratif.

5 Les codes JVM produits par les compilateurs Java existants satisfont à différents critères, lesquels sont énoncés ci-après :

C1 : les arguments de chaque instruction appartiennent bien aux types attendus par cette instruction ;

10 C2 : la pile ne déborde pas ;

C'3 : pour chaque instruction de branchement, le type de la pile au niveau de ce branchement est le même qu'au niveau des cibles possibles pour ce branchement ;

15 C'4 : une valeur de type t écrite dans un registre en un point du code et relue depuis ce même registre en un autre point du code est toujours relue avec le même type t ;

20 La mise en œuvre du procédé de vérification objet de la présente invention, implique que les critères C'3 et C'4 vérifiés par le code objet soumis à vérification soient remplacés par les critères C3 et C4 ci-après :

C3 : la pile est vide à chaque instruction de branchement et à chaque cible de branchement ;

25 C4 : un même registre est utilisé avec un seul et même type dans tout le code d'un sous-programme.

30 En référence aux critères précités, on indique que les compilateurs Java garantissent seulement les critères plus faibles C'3 et C'4, le processus de vérification objet de la présente invention et le protocole de gestion correspondant garantissent en fait des critères C3 et C4

plus contraignants permettant d'assurer la sécurité d'exécution et de gestion des appliquettes.

La notion de normalisation recouvrant la transformation des codes en codes normalisés peut
5 présenter différents aspects, dans la mesure où le remplacement des critères C'3 et C'4 par les critères C3 et C4, conformément au processus de vérification objet de la présente invention, peut être réalisé de manière indépendante pour assurer que la pile est vide à chaque
10 instruction de branchement et à chaque cible de branchement, respectivement que les registres ouverts par l'appliquette sont typés, à chaque registre ouvert correspondant un seul type de donnée attribué pour l'exécution de l'appliquette considérée, ou, au contraire,
15 de manière conjointe, afin de satisfaire à l'ensemble du processus de vérification objet de la présente invention.

Le procédé de transformation d'un code objet en code objet normalisé selon l'invention sera en conséquence décrit selon deux modes de mise en œuvre distincts, un
20 premier mode de mise en œuvre correspondant à la transformation d'un code objet satisfaisant aux critères C1, C2, C'3, C'4 en un code objet normalisé satisfaisant aux critères C1, C2, C3, C'4 correspondant à un code normalisé à instruction de branchement ou cible de
25 branchement vide, puis, selon un deuxième mode de réalisation, dans lequel le code objet classique satisfaisant aux mêmes critères de départ, est transformé en un code objet normalisé satisfaisant aux critères C1, C2, C'3, C4 par exemple correspondant à un code normalisé
30 faisant appel à des registres typés.

Le premier mode de réalisation du procédé de transformation de code, objet de la présente invention, sera maintenant décrit en liaison avec la figure 4a. Dans le mode de réalisation illustré en figure 4a, le code classique de départ est réputé satisfaire aux critères $C1+C2+C3$ et le code normalisé obtenu du fait de la transformation est réputé satisfaire aux critères $C1+C2+C3$.

Selon la figure précitée, le procédé de transformation consiste, pour chaque instruction courante I_i du code ou du sous-programme, à annoter chaque instruction, en une étape 500, par le type de données de la pile avant et après l'exécution de cette instruction. Les données d'annotation sont notées AI_i et sont associées par la relation $I_i \leftrightarrow AI_i$ en instruction courante considérée. Les données d'annotation sont calculées au moyen d'une analyse du flot des données relatif à cette instruction. Les types de données avant et après exécution de l'instruction sont notés tbe_i et tae_i respectivement. Le calcul des données d'annotation par analyse du flot des données est un calcul classique connu de l'homme du métier et, à ce titre, ne sera pas décrit en détail.

L'opération réalisée à l'étape 500 est illustrée au tableau T6 introduit en annexe dans lequel, pour une appliquette ou sous-programme d'appliquette comportant 12 instructions, les données d'annotation AI_i constituées par les types des registres et les types de la pile sont introduites.

L'étape 500 précitée est alors suivie d'une étape 500a consistant à positionner l'index i sur la première instruction $I_i = I_1$. L'étape 500a est suivie d'une étape 501

consistant à détecter, au sein des instructions et de chaque instruction courante I_i , l'existence de branchements notés IB ou de cibles de branchement CIB pour lesquels la pile d'exécution n'est pas vide. Cette
5 détection 501 est réalisée par un test conduit à partir des données d'annotation AI_i du type des variables de pile alloué à chaque instruction courante, le test étant noté pour l'instruction courante :

I_i est une IB ou CIB et $\text{pile}(AI) \neq \text{vide}$.

10 Sur réponse positive au test 501, c'est-à-dire en présence d'une détection d'une pile d'exécution non vide, le test précité est suivi d'une étape consistant à insérer des instructions de transfert des variables de pile de
15 part et d'autre de ces branchements IB ou de ces cibles de branchement CIB, afin de vider le contenu de la pile d'exécution dans des registres temporaires avant ce branchement et de rétablir la pile d'exécution à partir des registres temporaires après ce branchement. L'étape
20 d'insertion est notée 502 sur la figure 4a. Elle est suivie d'une étape de test 503 d'atteinte de la dernière instruction noté

$I_i = \text{dernière instruction?}$

Sur réponse négative au test 503, une incrémentation 504 $i=i+1$ est effectuée pour passage à l'instruction suivante
25 et retour à l'étape 501. Sur réponse positive au test 503, une étape de Fin est lancée. Sur réponse négative au test 501, le procédé de transformation est poursuivi par un branchement vers l'étape 503 en l'absence d'insertion d'instruction de transfert. La mise en œuvre du procédé de
30 transformation d'un code classique en un code normalisé à instruction de branchement à pile vide tel que représenté

en figure 4a, permet d'obtenir un code objet normalisé pour le même fragment de programme de départ dans lequel la pile des variables de pile est vide à chaque instruction de branchement et à chaque instruction de cible de branchement, en l'absence de modification de l'exécution du fragment de programme. Dans le cas d'un environnement Java, les instructions de transfert de données entre pile et registre sont les instructions load et store de la machine virtuelle Java.

En reprenant l'exemple introduit au tableau T6, le procédé de transformation détecte une cible de branchement où la pile n'est pas vide au niveau de l'instruction 9. Il est donc procédé à l'insertion d'une instruction istore 1 avant l'instruction de branchement 5 qui mène à l'instruction 9 précitée afin de sauvegarder le contenu de la pile dans le registre 1 et d'assurer que la pile est vide lors du branchement. Symétriquement, l'insertion d'une instruction iload 1 est effectuée préalablement à la cible d'instruction 9 pour rétablir le contenu de la pile identiquement à ce qu'il était avant le branchement. Finalement, une instruction istore 1 est insérée après l'instruction 8 pour garantir l'équilibre de la pile sur les deux chemins qui mènent à l'instruction 9. Le résultat de la transformation ainsi opérée en un code normalisé est représenté au tableau T7.

Le deuxième mode de réalisation du procédé de transformation objet de la présente invention sera maintenant décrit en liaison avec la figure 4b dans le cas où le code objet classique de départ satisfait aux critères C1+C'4 et le code objet normalisé satisfait aux critères C1+C4.

En référence à la figure 4b précitée, on indique que le procédé, dans ce mode de réalisation, consiste à annoter, selon une étape 500 sensiblement identique à celle représentée en figure 4a, chaque instruction courante I_i par le type de données des registres avant et après l'exécution de cette instruction. De la même manière, les données d'annotations AI_i sont calculées au moyen d'une analyse du flot des données relatif à cette instruction.

L'étape d'annotation 500 est alors suivie d'une étape consistant à effectuer une réallocation des registres, étape notée 601, par détection des registres d'origine employés avec des types différents, division de ces registres d'origine en registres normalisés distincts, un registre normalisé étant alloué à chaque type de données utilisé. L'étape 601 est suivie d'une étape 602 de réactualisation des instructions qui manipulent les opérandes qui font appel aux registres normalisés précités. L'étape 602 est suivie d'une étape de suite de contexte 302.

En référence à l'exemple donné au tableau T6, on indique que le procédé de transformation détecte que le registre de rang 0, noté $r0$, est utilisé avec les deux types object, instructions 0 et 1, et int, instruction 9 et suivantes. Il est alors procédé à une division du registre d'origine $r0$ en deux registres, le registre 0 pour l'utilisation des types object et le registre 1 pour les utilisations de type int. Les références au registre 0 de type int sont alors réécrites en les transformant en des références au registre 1, le code normalisé obtenu étant introduit au tableau T8 joint en annexe.

On note de manière non limitative que dans l'exemple introduit en liaison avec le tableau T8 précité, le nouveau registre 1 est utilisé à la fois pour la normalisation de la pile et pour la création de registres typés par division du registre 0 en deux registres.

Le procédé de transformation d'un code classique en un code normalisé à instruction de branchement à pile vide tel que décrit en figure 4a sera maintenant décrit de manière plus détaillée dans un mode de réalisation préférentiel non limitatif, en relation avec la figure 5a.

Ce mode de réalisation concerne l'étape 501 consistant à détecter au sein des instructions et de chaque instruction courante I_i l'existence de branchement IB respectivement de cible de branchement CIB pour laquelle la pile n'est pas vide.

Suite à la détermination des instructions cible où la pile n'est pas vide, cette condition étant notée à l'étape 504a, $I_i \text{ pile} \neq \text{vide}$, le processus de transformation consiste à associer, à l'étape 504a précitée, à ces instructions un ensemble de nouveaux registres, un par emplacement de pile actif au niveau de ces instructions. Ainsi, si i désigne le rang d'une cible de branchement dont le type de pile associée n'est pas vide et est du type tpl_i à tpn_i avec $n > 0$, pile non vide, le processus de transformation alloue n nouveau registres, r_1 à r_n non encore utilisés et les associe à l'instruction i correspondante. Cette opération est réalisée à l'étape 504a.

L'étape 504a est suivie d'une étape 504 consistant à examiner chaque instruction détectée de rang i et à discriminer en une étape de test 504 l'existence d'une

cible de branchement CIB ou d'un branchement IB. L'étape 504 est représentée sous forme d'un test désigné par :

$\exists ?CIB, IB \text{ et } I_i = CIB.$

5 Dans le cas où l'instruction de rang i est une cible de branchement CIB représentée par l'égalité précédente, et que la pile des variables de pile au niveau de cette instruction n'est pas vide, c'est-à-dire en réponse positive au test 504, pour toute instruction précédente de rang $i-1$ constituée par un branchement, une
10 levée d'exceptions ou un retour de programme, cette condition est réalisée à l'étape de test 505 désignée par :

$I_{i-1} = IB$, levée EXCEPT, retour Prog.

15 L'instruction détectée de rang i n'est accessible que par un branchement. Sur réponse positive au test 505 précité, le processus de transformation consiste à effectuer une étape 506 consistant à insérer un ensemble d'instructions de chargement du type load à partir de l'ensemble de nouveaux registres antérieurement à l'instruction détectée
20 de rang i considéré. L'opération d'insertion 506 est suivie d'une redirection 507 de tous les branchements vers l'instruction détectée de rang i , vers la première instruction de chargement load insérée. Les opérations d'insertion et de redirection sont représentées au tableau
25 T9 joint en annexe.

Pour toute instruction précédente de rang $i-1$ continuant en séquence, c'est-à-dire lorsque l'instruction courante de rang i est accessible à la fois par un branchement et à partir de l'instruction précédente, cette
30 condition étant réalisée par le test 508 et symbolisée par les relations :

$$I_{i-1} \rightarrow I_i$$

et

$$IB \rightarrow I_i$$

le processus de transformation consiste en une étape 509 à
5 insérer un ensemble d'instructions de sauvegarde store
vers l'ensemble de nouveaux registres antérieurement à
l'instruction détectée de rang i , et un ensemble
d'instructions de chargement load à partir de cet ensemble
de nouveaux registres. L'étape 509 est alors suivie d'une
10 étape 510 de redirection de tous les branchements vers
l'instruction détectée de rang i vers la première
instruction de chargement load insérée.

Dans le cas où l'instruction détectée de rang i
est un branchement vers une instruction déterminée, pour
15 toute instruction détectée de rang i constituée par un
branchement inconditionnel, cette condition étant réalisée
par un test 511 noté :

$$I_i = IB_{\text{incondit.}}$$

le processus de transformation tel que représenté en
20 figure 5a consiste à insérer à une étape 512, sur réponse
positive au test 511, antérieurement à l'instruction
détectée de rang i , une pluralité d'instructions de
sauvegarde store. Le processus de transformation insère
avant l'instruction i les n instructions store ainsi que
25 représenté en exemple au tableau T11. Les instructions
store adressent les registres r_1 à r_n , n désignant le
nombre de registres. A chaque nouveau registre est ainsi
associée l'instruction de sauvegarde.

Pour toute instruction détectée de rang i
30 constituée par un branchement conditionnel et pour un
nombre m_{Op} supérieur à 0 d'opérandes manipulés par cette

instruction de branchement conditionnel, cette condition étant réalisée par le test 513 noté :

$$I_i = IB_{\text{condit.}}$$

avec $mOp > 0$

5 le processus de transformation, en réponse positive au test 513 précité, consiste à insérer à une étape 514 antérieurement à cette instruction détectée de rang i , une instruction de permutation notée swap_x au sommet de la pile des variables de pile des mOp opérandes de l'instruction détectée de rang i et des n valeurs
10 suivantes. Cette opération de permutation permet de ramener au sommet de la pile des variables de pile les n valeurs à sauvegarder dans l'ensemble des nouveaux registres r_1 à r_n . L'étape 514 est suivie d'une étape 515 consistant à insérer antérieurement à l'instruction de rang i un ensemble d'instructions de sauvegarde store vers l'ensemble des nouveaux registres r_1 à r_n . L'étape 515 d'insertion précitée est elle-même suivie d'une étape 516 d'insertion postérieurement à l'instruction détectée de
20 rang i d'un ensemble d'instructions de chargement load à partir de l'ensemble des nouveaux registres r_1 à r_n . L'ensemble des opérations d'insertion correspondantes est représenté au tableau 12 introduit en annexe.

Pour des raisons de complétude et en référence à
25 la figure 5a, on indique que, sur réponse négative au test 504, la poursuite du processus de transformation est réalisée par une étape de suite de contexte suite 503, que la réponse négative aux tests, 505, 508, 511 et 513 est elle-même suivie d'une poursuite du processus de
30 transformation par l'intermédiaire d'une étape de suite de contexte suite 503 et qu'il en est de même en ce qui

concerne la poursuite des opérations après les étapes de redirection 507 et 510 et d'insertion 512 et 516 précitées.

5 Une description plus détaillée du procédé de normalisation et de transformation d'un code objet en un code objet normalisé faisant appel à des registres typés tel que décrit en figure 4b, sera maintenant donnée en liaison avec la figure 5b. Ce mode de réalisation concerne plus particulièrement un mode de réalisation préférentiel
10 non limitatif de l'étape 601 de réallocation des registres par détection des registres d'origine employés avec des types différents.

En référence à la figure 5b précitée, on indique que l'étape 601 précitée consiste à déterminer en une
15 étape 603 les intervalles de durée de vie notés ID_j de chaque registre r_j . Ces intervalles de durée de vie, désignés par "live range" ou "webs" en langage anglo-saxon, sont définis pour un registre r comme un ensemble maximal de traces partielles tel que le registre r est
20 vivant en tous points de ces traces. Pour une définition plus détaillée de ces notions, on pourra utilement se reporter à l'ouvrage édité par Steven S. MUCHNICK intitulé "Advanced Compiler Design and Implementation", Section 16.3, Morgan KAUFMANN, 1997. L'étape 603 est désignée par
25 la relation :

$$ID_j \leftrightarrow r_j$$

selon laquelle à chaque registre r_j est associé un intervalle de durée de vie ID_j correspondant.

L'étape 603 précitée est suivie d'une étape 604
30 consistant à déterminer, à l'étape 604, le type de données principal, noté tp_j , de chaque intervalle de durée de vie

5 ID_j. Le type principal d'un intervalle de durée de vie ID_j, pour un registre r_j, est défini par la borne supérieure des types de donnée stockés dans ce registre r_j par les instruction de sauvegarde store appartenant à l'intervalle de durée de vie précité.

L'étape 604 est elle-même suivie d'une étape 605 consistant à établir un graphe d'interférences entre les intervalles de durée de vie précédemment définis aux étapes 603 et 604, ce graphe d'interférences consistant en un graphe non orienté dont chaque sommet est constitué par un intervalle de durée de vie et dont les arcs, notés a_{j1,j2} sur la figure 5b, entre deux sommets ID_{j1} et ID_{j2}, existent si un sommet contient une instruction de sauvegarde adressée au registre de l'autre sommet ou réciproquement. Sur la figure 5b, la construction du graphe d'interférences est représentée symboliquement, cette construction pouvant être réalisée à partir de techniques de calcul connues de l'homme du métier. Pour une description plus détaillée de la construction de ce type de graphe, on pourra utilement se reporter à l'ouvrage publié par Alfred V.AHO, Ravi SETHI et Jeffrey D. ULLMAN intitulé "Compilers : principes, techniques, and tools", Addison-Wesley 1986, section 9.7.

25 Suite à l'étape 605, le procédé de normalisation tel que représenté en figure 5b consiste à traduire à une étape 606 l'unicité d'un type de donnée alloué à chaque registre r_j dans le graphe d'interférences en ajoutant des arcs entre toutes paires de sommets du graphe d'interférences tant que deux sommets d'une paire de sommets n'ont pas le même type de données principal associé. On comprend que la traduction du caractère

d'unicité d'un type de donnée alloué à chaque registre correspond bien entendu à la traduction et à la prise en compte du critère C4 dans le graphe d'interférences, critère mentionné précédemment dans la description.

5 L'étape 606 précitée est alors suivie d'une étape 607 dans laquelle une instanciation du graphe d'interférences est effectuée, instanciation plus communément désignée par étape de coloriage du graphe d'interférences selon les techniques habituelles. Au cours de l'étape 607, le

10 processus de transformation attribue à chaque intervalle de durée de vie ID_{jk} un numéro de registre rk de telle manière que deux intervalles adjacents dans le graphe d'interférences reçoivent des numéros de registres différents.

15 Cette opération peut être réalisée à partir de tout processus adapté. A titre d'exemple non limitatif, on indique qu'un processus préférentiel peut consister :

a) à choisir un sommet de degré minimal dans le graphe d'interférences, le degré minimal étant défini comme

20 un nombre de sommets adjacents minimal, et de le retirer du graphe. Cette étape peut être répétée jusqu'à ce que le graphe soit vide.

b) Chaque sommet précédemment retiré est réintroduit dans le graphe d'interférences dans l'ordre inverse de leur retrait, le dernier enlevé étant le premier réintroduit et successivement dans l'ordre inverse de l'ordre de retrait. Ainsi, à chaque sommet réintroduit, peut être attribué le plus petit numéro

25 de registre qui est différent des numéros attribués à tous les sommets adjacents.

30

Enfin, par l'étape 602, représentée en figure 4b, le processus de transformation et de réallocation réécrit les instructions d'accès aux registres figurant dans le code du sous-programme de l'appliquette considérée. Un accès à un registre donné dans l'intervalle de durée de vie correspondant est remplacé par un accès à un registre différent dont le numéro a été attribué pendant la phase d'instanciation encore désignée par phase de coloriage.

Une description plus détaillée d'un système informatique embarqué permettant la mise en œuvre du protocole de gestion et du processus de vérification d'un fragment de programme ou appliquette conforme à l'objet de la présente invention et d'un système de développement d'une appliquette sera maintenant donnée en liaison avec la figure 6.

En ce qui concerne le système embarqué correspondant portant la référence 10, on rappelle que ce système embarqué est un système du type reprogrammable comportant les éléments essentiels tels que représentés en figure 1b. Le système embarqué précité est réputé interconnecté à un terminal par une liaison série, le terminal étant lui-même relié par exemple par l'intermédiaire d'un réseau local, le cas échéant d'un réseau lointain, à un ordinateur de développement de l'appliquette portant la référence 20. Sur le système embarqué 10 fonctionne un programme principal qui lit et exécute les commandes envoyées sur la liaison série par le terminal. En outre, les commandes standard pour une carte à microprocesseur, telles que par exemple les commandes standard du protocole ISO 7816, peuvent être mises en œuvre, le programme principal reconnaissant de plus deux

commandes supplémentaires, l'une pour le téléchargement d'une appliquette, et l'autre pour la sélection d'une appliquette préalablement chargée sur la carte à microprocesseur.

5 Conformément à l'objet de la présente invention, la structure du programme principal est réalisée de manière à comporter au moins un module de programme de gestion et de vérification d'un fragment de programme téléchargé suivant le protocole de gestion d'un fragment
10 de programme téléchargé précédemment décrit dans la description en liaison avec la figure 2.

 En outre, le module de programme comporte également un module de sous-programme de vérification d'un fragment de programme téléchargé suivant le procédé de
15 vérification tel que décrit précédemment dans la description en liaison avec les figures 3a à 3j.

 Dans ce but, la structure des mémoires, en particulier de la mémoire permanente non inscriptible, mémoire ROM, est modifiée de manière à comporter
20 notamment, outre le programme principal, un module 17 de gestion de protocole et de vérification, ainsi que mentionné précédemment. Enfin, en ce qui concerne la mémoire non volatile réinscriptible de type EEPROM, celle-ci comporte avantageusement un répertoire d'appliquettes,
25 noté 18, permettant la mise en œuvre du protocole de gestion et du processus de vérification objets de la présente invention.

 En référence à la même figure 6, on indique que le système de développement de l'appliquette conforme à
30 l'objet de la présente invention, permettant en fait la transformation d'un code objet classique ainsi que

mentionné précédemment dans la description et satisfaisant aux critères C1+C2+C'3+C'4 dans le cadre de l'environnement Java en un code objet normalisé pour le même fragment de programme comprend, associé à un

5 compilateur classique Java 21, un module de transformation de code, noté 22, lequel procède à la transformation de code en code normalisé selon le premier et le deuxième mode de réalisation précédemment décrits dans la description en liaison avec les figures 4a, 4b et 5a, 5b.

10 On comprend en effet que, d'une part, la normalisation du code objet d'origine en un code objet normalisé à instruction de branchement à pile vide et en un code normalisé faisant appel à des registres typés, d'autre part, ainsi que mentionné précédemment dans la

15 description, permet de satisfaire aux critères de vérification C3 et C4 imposés par le procédé de vérification objet de la présente invention.

Le module de transformation de code 22 est suivi d'un convertisseur JavaCard 23, lequel permet d'assurer la

20 transmission par un réseau distant ou local vers le terminal et, par l'intermédiaire de la liaison série, vers la carte à microprocesseur 10. Ainsi, le système de développement de l'appliquette 20 représenté en figure 6 permet de transformer les fichiers de classe compilés

25 produits par le compilateur Java 21 à partir des codes source Java de l'appliquette en des fichiers de classe équivalents mais qui respectent les contraintes supplémentaires C3, C4 imposées par le protocole de gestion et le module de vérification 17 embarqués sur la

30 carte à microprocesseur 10. Ces fichiers de classe transformés sont convertis en une appliquette

téléchargeable sur la carte par le convertisseur JavaCard standard 23.

Différents éléments particulièrement remarquables de l'ensemble des éléments du protocole, des procédés et des systèmes objets de la présente invention, seront
5 maintenant donnés à titre indicatif.

Vis-à-vis des processus de vérification de l'art antérieur tels que mentionnés dans l'introduction à la description, le procédé de vérification objet de la présente invention apparaît remarquable en ce qu'il
10 concentre l'effort de vérification sur les propriétés de typage des opérandes qui sont essentielles à la sécurité de l'exécution de chaque applique, c'est-à-dire du respect des contraintes de type associées à chaque instruction et absence de débordement de pile. D'autres
15 vérifications n'apparaissent pas essentielles en termes de sécurité, en particulier la vérification que le code initialise correctement chaque registre avant de le lire pour la première fois. Le procédé de vérification objet de la présente invention opère au contraire par
20 initialisation à zéro de tous les registres à partir de la machine virtuelle lors de l'initialisation de la méthode afin de garantir que la lecture d'un registre non initialisé ne peut compromettre la sécurité de la carte.

En outre, l'exigence imposée par le procédé de
25 vérification objet de la présente invention selon laquelle la pile doit être vide à chaque instruction de branchement ou de cible de branchement, garantit que la pile est dans le même état, vide, après exécution du branchement et avant exécution de l'instruction à laquelle le programme
30 s'est branché. Ce mode opératoire garantit que la pile est

dans un état cohérent, quel que soit le chemin d'exécution suivi à travers le code du sous-programme ou de l'appliquette considéré. La cohérence de la pile est ainsi garantie, même en présence de branchement ou de cible de branchement. Contrairement aux procédés et aux systèmes de l'art antérieur, dans lesquels il est nécessaire de conserver en mémoire vive le type de la pile à chaque cible de branchement, ce qui nécessite une quantité de mémoire vive proportionnelle à $T_{px}Nb$, produit de la taille maximale de pile d'exécution utilisée et du nombre de cibles de branchement dans le code, le procédé de vérification, objet de la présente invention, n'a besoin que du type de la pile d'exécution lors de l'instruction en cours de vérification et il ne conserve pas en mémoire le type de cette pile à d'autres points du code. En conséquence, le procédé objet de l'invention se contente d'une quantité de mémoire vive proportionnelle à T_p mais indépendante de Nb , et par conséquent de la longueur du code du sous-programme ou de l'appliquette.

L'exigence selon le critère C4, selon lequel un registre donné doit être utilisé avec un seul et même type dans tout le code d'un sous-programme, garantit que le code précité n'utilise pas un registre de manière incohérente, par exemple en y écrivant un entier short à un point du programme et en le relisant comme une référence d'objet à un autre point du programme.

Dans les processus de vérification décrits dans l'art antérieur, en particulier dans la spécification Java intitulée "The Java Virtual Machine Specification" éditée par Tim LINDHOLM et Frank YELLIN, déjà citée, pour garantir la cohérence des utilisations précitées à travers

les instructions de branchement, il est nécessaire de conserver en mémoire vive une copie du tableau des types de registres à chaque cible de branchement. Cette opération nécessite une quantité de mémoire vive proportionnelle à $T_r \times N_b$ où T_r désigne le nombre de registres utilisés par le sous-programme et N_b le nombre de cibles de branchement dans le code de ce sous-programme.

Au contraire, le processus de vérification objet de la présente invention, opère sur un tableau global de types de registres en l'absence de conservation en mémoire vive de copie à différents points du code. En conséquence, la mémoire vive nécessaire pour réaliser le processus de vérification est proportionnelle à T_r mais indépendante de N_b et donc de la longueur du code du sous-programme considéré.

La contrainte selon laquelle un registre donné est utilisé avec le même type en tous points, c'est-à-dire en toute instruction du code considéré, simplifie sensiblement et de manière significative la vérification des sous-programmes. Au contraire, dans les processus de vérification de l'art antérieur, en l'absence d'une telle contrainte, le processus de vérification doit établir que les sous-programmes respectent une discipline de pile stricte et doit vérifier le corps des sous-programmes de manière polymorphe en ce qui concerne le type de certains registres.

En conclusion, le processus de vérification objet de la présente invention vis-à-vis des techniques de l'art antérieur permet, d'une part, de réduire la taille du code du programme permettant de conduire le procédé de

vérification, et, d'autre part, de réduire la consommation de mémoire vive lors des opérations de vérification, le degré de complexité étant de la forme $O(T_p + P_r)$ dans le cas du processus de vérification objet de la présente invention, au lieu de $(O(T_p + T_r) \times N_b)$ pour ce qui concerne les processus de vérification de l'art antérieur, en offrant toutefois les mêmes garanties vis-à-vis de la sûreté de l'exécution du code vérifié.

Enfin, le processus de transformation d'un code classique d'origine en un code normalisé est réalisé par transformation localisée du code en l'absence de transmission d'informations supplémentaires à l'organe vérificateur, c'est-à-dire à la carte à microprocesseur ou au système informatique embarqué.

En ce qui concerne le procédé de réallocation des registres tel que décrit en figures 4b et 5b, ce procédé se distingue des procédés connus de l'art antérieur décrits notamment par le brevet US 4,571,678 et par le brevet US 5,249,295, par le fait :

- que la réallocation de registre assure qu'un même registre ne peut être attribué à deux intervalles possédant des types principaux différents, ce qui garantit ainsi qu'un registre donné est utilisé avec le même type dans tout le code ; et
- que les algorithmes d'allocations de registres existants et décrits dans les documents précités supposent un nombre fixe de registres et tentent de minimiser les transferts désignés par "spills" en langage anglo-saxon, entre registres et pile, alors que la réallocation des registres conformément à l'objet de la présente invention opère dans un cadre où le nombre

total de registres est variable, en conséquence de quoi il n'y a pas lieu d'effectuer des transferts entre registres et piles alors qu'un processus de minimisation du nombre total de registres est mis en œuvre.

Le protocole de gestion d'un fragment de programme téléchargé sur un système embarqué et les procédés de vérification de ce fragment de programme téléchargé, respectivement de transformation de ce code objet de fragment de programme téléchargé, objets de la présente invention, peuvent bien entendu être mis en œuvre de manière logicielle.

A ce titre, l'invention concerne également un produit programme d'ordinateur chargeable directement dans la mémoire interne d'un système embarqué reprogrammable, ce système embarqué permettant le téléchargement d'un fragment de programme constitué par un code objet, suite d'instructions, exécutable par le microprocesseur du système embarqué par l'intermédiaire d'une machine virtuelle munie d'une pile d'exécution et de registres ou variables locales manipulés par ces instructions afin de permettre l'interprétation de ce code objet. Le produit programme d'ordinateur correspondant comprend des portions de code objet pour l'exécution du protocole de gestion d'un fragment de programme téléchargé sur ce système embarqué, ainsi qu'illustré sur la figure 2 et la figure 6 précédemment décrites dans la description, lorsque ce système embarqué est interconnecté à un terminal et que ce programme est exécuté par le microprocesseur de ce système embarqué par l'intermédiaire de la machine virtuelle.

L'invention concerne également un produit programme d'ordinateur chargeable directement dans la mémoire interne d'un système embarqué reprogrammable, tel qu'une carte à microprocesseur munie d'une mémoire réinscriptible, ainsi qu'illustré en liaison avec la figure 6. Ce produit programme d'ordinateur comprenant des portions de code objet pour l'exécution des étapes de vérification d'un fragment d'un programme téléchargé sur ce système embarqué, ainsi qu'illustré et décrit précédemment dans la description en liaison avec les figures 3a à 3j. Cette vérification est exécutée lorsque ce système embarqué est interconnecté à un terminal et que ce programme est exécuté par le microprocesseur de ce système embarqué par l'intermédiaire de la machine virtuelle.

L'invention concerne également un produit programme d'ordinateur ; ce produit programme d'ordinateur comprend des portions de code objet pour l'exécution des étapes du procédé de transformation du code objet d'un fragment de programme en un code objet normalisé pour ce même fragment de programme, ainsi qu'illustré et représenté aux figures 4a, 4b, respectivement 5a, 5b, ainsi qu'à la figure 6 et décrites précédemment dans la description.

La présente invention concerne également un produit programme d'ordinateur enregistré sur un support utilisable dans un système embarqué reprogrammable, une carte à microprocesseur munie d'une mémoire réinscriptible par exemple, ce système embarqué permettant le téléchargement d'un fragment de programme constitué par un code objet exécutable par ce microprocesseur, par

l'intermédiaire d'une machine virtuelle munie d'une pile d'exécution et de registres ou variables locales manipulés par ces instructions, afin de permettre l'interprétation de ce code objet. Le produit programme d'ordinateur précité comprend, au moins, un module de programmes lisibles par le microprocesseur du système embarqué par l'intermédiaire de la machine virtuelle, pour commander l'exécution d'une procédure de gestion du téléchargement d'un fragment de programme téléchargé, ainsi que représenté en figure 2 et décrit précédemment dans la description, un module de programmes lisibles par le microprocesseur par l'intermédiaire de la machine virtuelle pour commander l'exécution d'une procédure de vérification instruction par instruction du code objet constitutif du fragment de programme, ainsi qu'illustré et décrit en relation avec les figures 3a à 3j dans la description précédente, et un module de programmes lisibles par le microprocesseur de ce système embarqué par l'intermédiaire de la machine virtuelle pour commander l'exécution d'un fragment de programme téléchargé suite à ou en l'absence d'une transformation du code objet de ce fragment de programme en code objet normalisé pour ce même fragment de programme, ainsi que représenté en figure 2.

Le produit programme d'ordinateur précité comprend également un module de programmes lisibles par le microprocesseur par l'intermédiaire de la machine virtuelle pour commander l'inhibition de l'exécution, sur le système embarqué, du fragment de programme dans le cas d'une procédure de vérification non réussie de fragment de programme précité, ainsi qu'illustré et décrit

précédemment dans la description en liaison avec la figure 2.

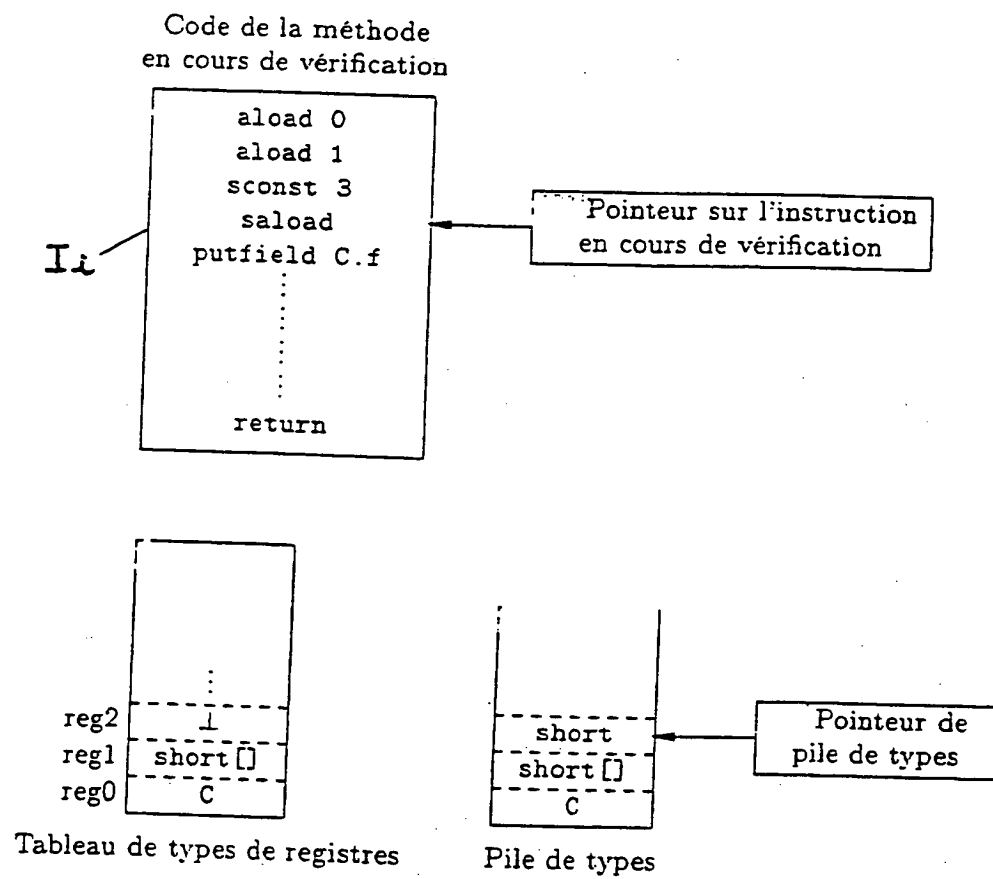
ANNEXESTABLEAU 1

TABLEAU 2Pseudo-code module du vérificateur.PSEUDO-CODE DU MODULE VERIFICATEUR

Variables globales utilisées:

T_r	nombre de registres déclaré par la méthode courante
T_p	taille maximale de pile déclarée par la méthode courante
$tr[T_r]$	tableau de types de registres (402 dans la figure 4)
$tp[T_p]$	type de pile (403 dans la figure 4)
pp	pointeur de pile (404 dans la figure 4)
chg	drapeau indiquant si tr a changé

Initialiser $pp \leftarrow 0$

Initialiser $tp[0] \dots tp[n-1]$ à partir des types des n arguments de la méthode

Initialiser $tp[n] \dots tp[T_r-1]$ à \perp

Initialiser chg à vrai

Tant que chg est vrai:

Remettre chg à faux

Se positionner sur la première instruction de la méthode

Tant que la fin de la méthode n'est pas atteinte:

Si l'instruction courante est la cible d'une instruction de branchement:

Si $pp \neq 0$, échec de la vérification

Si l'instruction courante est la cible d'un appel de sous-routine:

Si l'instruction précédente continue en séquence, échec

Prendre $tp[0] \leftarrow \text{retaddr}$ et $pp \leftarrow 1$

Si l'instruction courante est un gestionnaire d'exceptions de classe C :

Si l'instruction précédente continue en séquence, échec

Faire $tp[0] \leftarrow C$ et $pp \leftarrow 1$

Si l'instruction courante est une cible de sortes différentes:

Échec de la vérification

Déterminer les types a_1, \dots, a_n des arguments de l'instruction

Si $pp < n$, échec (débordement de pile)

Pour $i = 1, \dots, n$:

Si $tp[pp-n-i-1]$ n'est pas sous-type de a_i , échec

Faire $pp \leftarrow pp - n$

Déterminer les types r_1, \dots, r_m des résultats de l'instruction

Si $pp + m \geq T_p$, échec (débordement de pile)

Pour $i = 1, \dots, m$, faire $tp[pp+i-1] \leftarrow r_i$

Faire $pp \leftarrow pp + m$

Si l'instruction courante est une écriture dans un registre r :

Déterminer le type t de la valeur écrite dans le registre

Faire $tr[r] \leftarrow \text{borne inférieure}(t, tr[r])$

Si $tr[r]$ a changé, faire $chg \leftarrow \text{vrai}$

Si l'instruction courante est un branchement:

Si $pp \neq 0$, échec de la vérification

Avancer à la prochaine instruction

Renvoyer un code de succès de la vérification

TABLEAU T3

```

static short[] meth(short [] tableau)
{
    short[] resultat = null;
    if (tableau.length >= 2) resultat = tableau;
    return tableau;
}

```

TABLEAU T4

Première itération sur le code de la méthode:

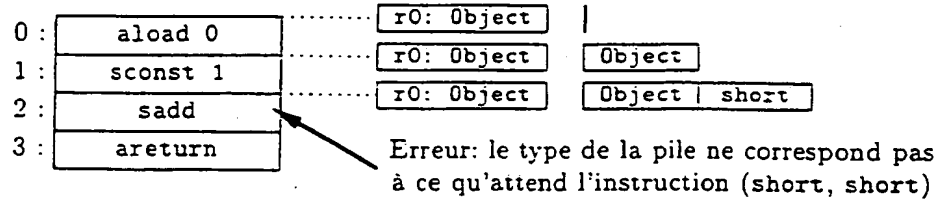
Code de la méthode	Tableau des types des registres	Type de la pile
0: aconst_null	r0: short[] r1: 1	
1: astore 1	r0: short[] r1: 1	null
2: aload 0	r0: short[] r1: null	
3: arraylength	r0: short[] r1: null	short[]
4: sconst 2	r0: short[] r1: null	short
5: if_scmlt 9	r0: short[] r1: null	short short
7: aload 0	r0: short[] r1: null	
8: astore 1	r0: short[] r1: null	short[]
9: aload 1	r0: short[] r1: short[]	
10: areturn	r0: short[] r1: short[]	short[]

Seconde itération sur le code de la méthode:

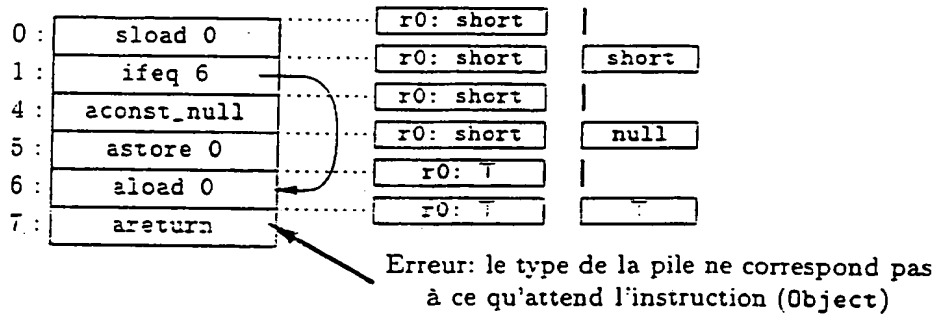
0: aconst_null	r0: short[] r1: short[]	
1: astore 1	r0: short[] r1: short[]	null
2: aload 0	r0: short[] r1: short[]	
3: arraylength	r0: short[] r1: short[]	short[]
4: sconst 2	r0: short[] r1: short[]	short
5: if_scmlt 9	r0: short[] r1: short[]	short short
7: aload 0	r0: short[] r1: short[]	
8: astore 1	r0: short[] r1: short[]	short[]
9: aload 1	r0: short[] r1: short[]	
10: areturn	r0: short[] r1: short[]	short[]

TABLEAU T5

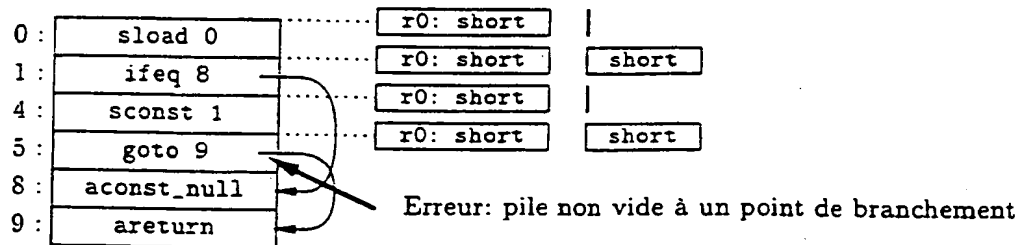
(a) *Violation des contraintes de types sur les arguments d'une instruction:*



(b) *Utilisation incohérente d'un registre:*



(c) *Branchements introduisant des incohérences au niveau de la pile:*



(d) *Débordement de pile à l'intérieur d'une boucle:*

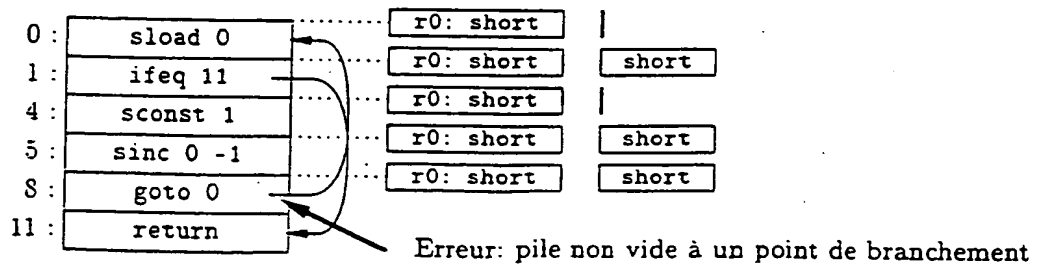


TABLEAU T6

(a) Code initial de la méthode, annoté par les types des registres et de la pile:

0:	aload 0	r0: Object	
1:	ifnull 8	r0: Object	Object
4:	iconst 1	r0: Object	
5:	goto 9	r0: Object	int
8:	iconst 0	r0: Object	
9:	ineg	r0: Object	int
10:	istore 0	r0: Object	int
11:	iload 0	r0: int	
12:	ireturn	r0: int	int

TABLEAU T7

(b) Code de la méthode après normalisation de la pile au niveau du branchement 5 → 9:

0:	aload 0	r0: Object	r1: ⊥	
1:	ifnull 8	r0: Object	r1: ⊥	Object
4:	iconst 1	r0: Object	r1: ⊥	
4':	istore 1	r0: Object	r1: ⊥	int
5:	goto 8''	r0: Object	r1: int	
8:	iconst 0	r0: Object	r1: int	
8':	istore 1	r0: Object	r1: ⊥	int
8'':	iload 1	r0: Object	r1: int	
9:	ineg	r0: Object	r1: int	int
10:	istore 0	r0: Object	r1: int	int
11:	iload 0	r0: int	r1: int	
12:	ireturn	r0: int	r1: int	int

TABLEAU T8

(c) Code de la méthode après réallocation des registres:

0:	aload 0	r0: Object	r1: ⊥	
1:	ifnull 8	r0: Object	r1: ⊥	Object
4:	iconst 1	r0: Object	r1: ⊥	
4':	istore 1	r0: Object	r1: ⊥	int
5:	goto 8''	r0: Object	r1: int	
8:	iconst 0	r0: Object	r1: int	
8':	istore 1	r0: Object	r1: ⊥	int
8'':	iload 1	r0: Object	r1: int	
9:	ineg	r0: Object	r1: int	int
10:	istore 1	r0: Object	r1: int	int
11:	iload 1	r0: Object	r1: int	
12:	ireturn	r0: Object	r1: int	int

TABLEAU T9

(a) Cible de branchement, instruction précédente ne continuant pas en séquence:

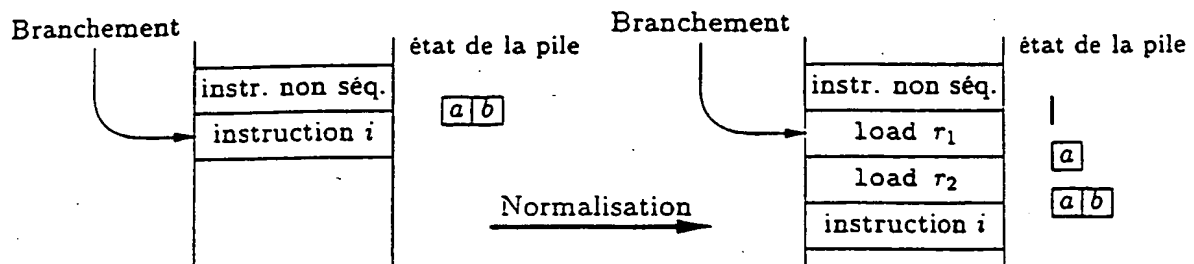
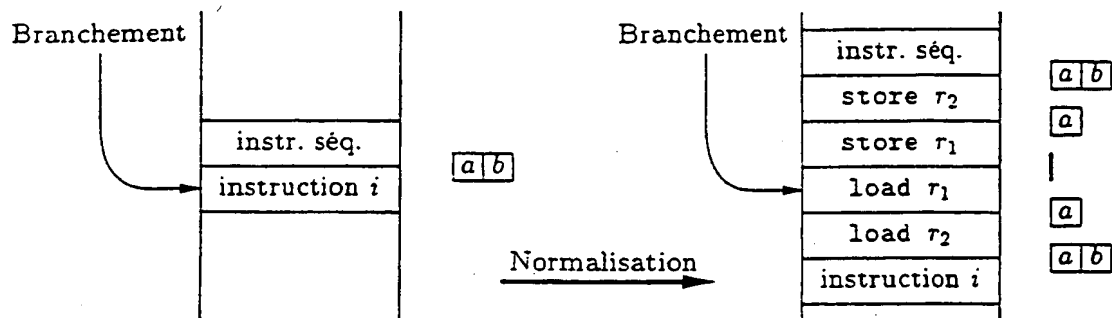
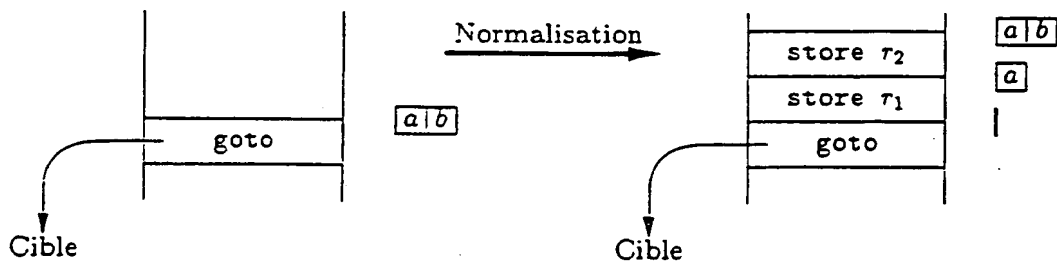
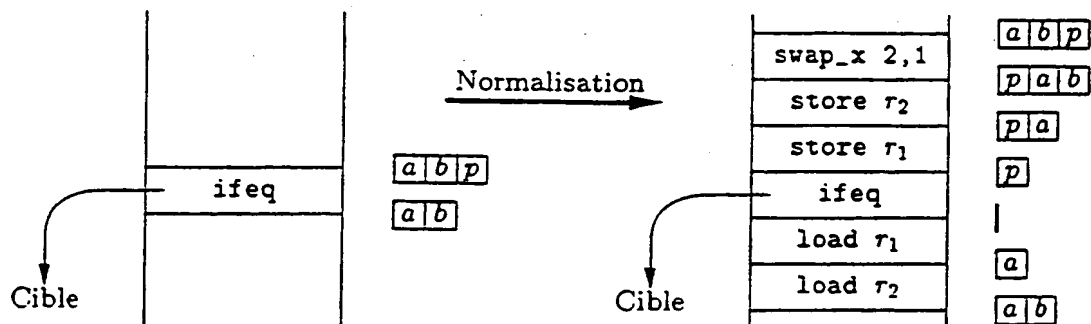


TABLEAU T10(b) *Cible de branchement, instruction précédente continuant en séquence:***TABLEAU T11**(c) *Branchement inconditionnel sans arguments:***TABLEAU T12**(d) *Branchement conditionnel à un argument:*

REVENDICATIONS

1. Protocole de gestion d'un fragment de programme téléchargé sur un système embarqué reprogrammable, tel qu'une carte à microprocesseur munie d'une mémoire réinscriptible, ledit fragment de programme étant constitué par un code objet, suite d'instructions, exécutable par le microprocesseur du système embarqué par l'intermédiaire d'une machine virtuelle munie d'une pile d'exécution et de registres ou variables locales manipulées par ces instructions et permettant d'interpréter ce code objet, ledit système embarqué étant interconnecté à un terminal, caractérisé en ce que ce protocole consiste au moins, au niveau dudit système embarqué :
- a) à détecter une commande de téléchargement de ce fragment de programme ; et sur réponse positive à cette étape consistant à détecter une commande de téléchargement,
 - b) à lire le code objet constitutif de ce fragment de programme et à mémoriser temporairement ce code objet ;
 - c) à soumettre l'ensemble du code objet mémorisé temporairement à un processus de vérification instruction par instruction, ce processus de vérification consistant au moins en une étape d'initialisation de la pile des types et du tableau des types de registres, représentant l'état de ladite machine virtuelle au début de l'exécution du code objet mémorisé temporairement et en une succession d'étapes de vérification instruction par instruction, par discrimination de l'existence, pour chaque

instruction courante, d'une cible, cible d'instruction de branchement, cible d'un appel d'un gestionnaire d'exceptions ou cible d'un appel de sous-routine et par une vérification et une actualisation de l'effet de ladite instruction courante sur la pile des types et sur le tableau des types de registres, et, dans le cas d'une vérification réussie dudit code objet,

- d) à enregistrer le fragment de programme téléchargé dans un répertoire de fragments de programmes disponibles, et, dans le cas d'une vérification non réussie dudit code objet,
- e) à inhiber l'exécution, sur ledit système embarqué, dudit fragment de programme.

2. Protocole selon la revendication 1, caractérisé en ce que ladite étape e) d'inhibition de l'exécution consiste :

- f) à effacer le fragment de programme enregistré momentanément, en l'absence d'enregistrement de ce dernier dans ledit répertoire de fragments de programmes disponibles, et
- g) à adresser audit lecteur un code d'erreur.

3. Protocole selon la revendication 1 ou 2, caractérisé en ce que, sur réponse négative à ladite étape a) consistant à détecter une commande de téléchargement, celui-ci consiste :

- b') à détecter une commande de sélection d'un fragment de programme disponible dans un répertoire de fragments de programmes ; et, sur réponse positive à cette étape consistant à détecter une commande de sélection d'un fragment de programme disponible ;

c') à appeler ledit fragment de programme disponible sélectionné ;

d') à exécuter ledit fragment de programme disponible appelé par l'intermédiaire de la machine virtuelle, en l'absence de toute vérification dynamique de types de variables, des droits d'accès aux objets manipulés par le fragment de programme disponible appelé, du débordement de la pile d'exécution lors de l'exécution de chaque instruction, et, sur réponse négative à cette étape consistant à détecter une commande de sélection d'un fragment de programme disponible,

e') à procéder au traitement des commandes standards du système embarqué.

4. Procédé de vérification d'un fragment de programme téléchargé sur un système embarqué reprogrammable, tel qu'une carte à microprocesseur munie d'une mémoire réinscriptible, ledit fragment de programme étant constitué par un code objet et comportant au moins un sous-programme, suite d'instructions, exécutable par le microprocesseur du système embarqué par l'intermédiaire d'une machine virtuelle munie d'une pile d'exécution et de registres d'opérandes manipulés par ces instructions et permettant d'interpréter ce code objet, ledit système embarqué étant interconnecté à un lecteur, caractérisé en ce que ledit procédé consiste, suite à la détection d'une commande de téléchargement et à la mémorisation dudit code objet constitutif de ce fragment de programme dans ladite mémoire réinscriptible, pour chaque sous-programme :

α) à effectuer une étape d'initialisation de la pile des types et du tableau des types de registres par des données représentant l'état de la machine virtuelle au

début de l'exécution du code objet mémorisé temporairement ;

- 5 β) à effectuer une vérification dudit code objet mémorisé temporairement instruction par instruction, par discrimination de l'existence, pour chaque instruction courante, d'une cible, cible d'instruction de branchement, cible d'un appel d'un gestionnaire d'exceptions ou cible d'un appel de sous-routine ;
- 10 γ) à effectuer une vérification et une actualisation de l'effet de ladite instruction courante sur les types de données de ladite pile des types et dudit tableau des types de registres, en fonction de l'existence d'une cible d'instruction de branchement, d'une cible d'un appel de sous-routine ou d'une cible d'un appel
- 15 de gestionnaire d'exceptions, ladite vérification étant réussie lorsque le tableau des types de registres n'est pas modifié au cours d'une vérification de toutes les instructions et le processus de vérification étant poursuivi instruction
- 20 par instruction jusqu'à ce que le tableau des types de registres soit stable, en l'absence de modification, le processus de vérification étant interrompu sinon.

5. Procédé de vérification selon la revendication 4, caractérisé en ce que les types de variables manipulées

25 au cours du processus de vérification comprennent au moins :

- des identificateurs de classes correspondant aux classes d'objets définies dans le fragment de
- 30 programme ;
- des types de variables numériques comportant au moins un type short, entier codé sur p bits, et un type

retaddr d'adresse de retour d'une instruction de saut JSR ;

- un type null relatif à des références d'objets nulles ;
- un type object relatif aux objets ;
- 5 - un premier type spécifique ⊥, représentant l'intersection de tous les types et correspondant à la valeur 0, nil ;
- un deuxième type spécifique T, représentant l'union de tous les types et correspondant à tout type de valeur.

10 6. Procédé selon la revendication 5, caractérisé en ce que l'ensemble desdits types de variables vérifie une relation de sous-typage :

object \in T;

short, retaddr \in T;

15 $\perp \in$ null, short, retaddr.

7. Procédé selon l'une des revendications 4 à 6, caractérisé en ce que lorsque ladite instruction courante est la cible d'une instruction de branchement, ledit procédé de vérification consiste à vérifier que la pile
20 des types est vide, le processus de vérification étant poursuivi pour l'instruction suivante dans le cas d'une vérification positive, et, le processus de vérification échouant et le fragment de programme étant rejeté sinon.

8. Procédé selon l'une des revendications 4 à 7,
25 caractérisé en ce que lorsque ladite instruction courante est la cible d'un appel de sous-routine, ledit processus de vérification vérifie que l'instruction précédente constitue un branchement inconditionnel, un retour de sous-routine ou une levée d'exception, ledit processus de
30 vérification, en cas de vérification positive, procédant à

une réactualisation de la pile des types de variables par une entité de type retaddr, adresse de retour de la sous-routine, et, le processus de vérification échouant et le fragment de programme étant rejeté sinon.

5 9. Procédé selon l'une des revendications 4 à 8, caractérisé en ce que lorsque l'instruction courante est la cible d'un gestionnaire d'exceptions, ledit processus de vérification vérifie que l'instruction précédente constitue un branchement inconditionnel, un retour de
10 sous-routine ou une levée d'exception, ledit processus de vérification, en cas de vérification positive, procédant à une réactualisation de la pile des types par une entrée du type des exceptions, et, le processus de vérification échouant et le fragment de programme étant rejeté sinon.

15 10. Procédé selon l'une des revendications 4 à 9, caractérisé en ce que lorsque l'instruction courante est la cible d'une pluralité de branchements incompatibles, le processus de vérification échoue et le fragment de programme est rejeté.

20 11. Procédé selon l'une des revendications 4 à 10, caractérisé en ce que lorsque l'instruction courante n'est la cible d'aucun branchement, le processus de vérification continue par passage à une réactualisation de la pile des types.

25 12. Procédé selon l'une des revendications 4 à 11, caractérisé en ce que l'étape de vérification de l'effet de l'instruction courante sur la pile des types comporte au moins :

- une étape de vérification que la pile d'exécution des
30 types contient au moins autant d'entrées que l'instruction courante comporte d'opérandes ;

- une étape de dépilement et de vérification que les types des entrées au sommet de la pile sont sous-types des types des opérandes de cette instruction ;
- une étape de vérification de l'existence d'un espace mémoire suffisant sur la pile des types pour procéder à l'empilement des résultats de l'instruction courante ;
- une étape d'empilement sur la pile des types de données attribués à ces résultats.

13. Procédé selon la revendication 12, caractérisé en ce que lorsque l'instruction courante est une instruction de lecture d'un registre d'adresse n, le processus de vérification consiste :

- à vérifier le type de donnée du résultat de cette lecture par consultation de l'entrée n du tableau des types de registres ;
- à déterminer l'effet de l'instruction courante sur la pile des types par dépilement des entrées de la pile correspondant aux opérandes de cette instruction courante et par empilement du type de données de ce résultat.

14. Procédé selon la revendication 12, caractérisé en ce que lorsque l'instruction courante est une instruction d'écriture d'un registre d'adresse m, le processus de vérification consiste :

- à déterminer l'effet de l'instruction courante sur la pile des types et le type t de l'opérande écrit dans ce registre d'adresse m ;
- à remplacer l'entrée de type du tableau des types de registres à l'adresse m par le type immédiatement supérieur au type précédemment stocké et au type t de l'opérande écrit dans ce registre d'adresse m.

15. Procédé de transformation d'un code objet d'un fragment de programme, dans lequel les opérandes de chaque instruction appartiennent aux types de données manipulées par cette instruction, la pile d'exécution ne présente pas
- 5 de phénomène de débordement, pour chaque instruction de branchement le type des variables de pile au niveau de ce branchement est le même qu'au niveau des cibles de ce branchement, en un code objet normalisé pour ce même fragment de programme, dans lequel les opérandes de chaque
- 10 instruction appartiennent aux types de données manipulées par cette instruction, la pile d'exécution ne présente pas de phénomène de débordement, la pile d'exécution est vide à chaque instruction de branchement et à chaque instruction de cible de branchement, caractérisé en ce que
- 15 ce procédé consiste, pour l'ensemble des instructions dudit code objet :
- à annoter chaque instruction courante par le type de données de la pile avant et après l'exécution de cette instruction, les données d'annotation étant calculées
 - 20 au moyen d'une analyse du flot des données relatif à cette instruction ;
 - à détecter au sein desdites instructions et de chaque instruction courante l'existence de branchements respectivement de cibles de branchements pour lesquels
 - 25 ladite pile d'exécution n'est pas vide, l'opération de détection étant conduite à partir des données d'annotation du type des variables de pile allouées à chaque instruction courante, et en présence d'une détection d'une pile d'exécution non vide,
 - 30 - à insérer des instructions de transfert des variables de pile de part et d'autre de ces branchements

respectivement de ces cibles de branchements afin de vider le contenu de la pile d'exécution dans des registres temporaires avant ce branchement et de rétablir la pile d'exécution à partir desdits registres temporaires après ce branchement, et à n'insérer aucune instruction de transfert sinon, ce qui permet d'obtenir un code objet normalisé pour ce même fragment de programme, dans lequel la pile d'exécution est vide à chaque instruction de branchement et à chaque instruction de cible de branchement, en l'absence de modification de l'exécution dudit fragment de programme. ;

16. Procédé de transformation d'un code objet d'un fragment de programme, dans lequel les opérandes de chaque instructions appartiennent aux types de données manipulées par cette instruction, et un opérande de type déterminé écrit dans un registre par une instruction de ce code objet est relu depuis ce même registre par une autre instruction de ce code objet avec le même type de donnée déterminé, en un code objet normalisé pour ce même fragment de programme, dans lequel les opérandes de chaque instruction appartiennent aux types de données manipulées par cette instruction, un seul et même type de donnée étant alloué à un même registre dans tout ledit code objet normalisé, caractérisé en ce que ce procédé consiste, pour l'ensemble des instructions dudit code objet :

- à annoter chaque instruction courante par le type de donnée des registres avant et après l'exécution de cette instruction, les données d'annotation étant calculées au moyen d'une analyse du flot des données relatif à cette instruction ;

- à effectuer une réallocation des registres par détection des registres d'origine employés avec des types différents, division de ces registres d'origine en registres normalisés distincts, un registre normalisé pour chaque type de donnée utilisé, et une réactualisation des instructions qui manipulent les opérandes qui font appel auxdits registres normalisés.

17. Procédé selon la revendication 15, caractérisé en ce que l'étape consistant à détecter au sein desdites instructions et de chaque instruction courante l'existence de branchements respectivement de cibles de branchement pour laquelle la pile d'exécution n'est pas vide consiste, suite à la détection de chaque instruction de rang i correspondant :

- à associer à chaque instruction de rang i un ensemble de nouveaux registres, un nouveau registre étant associé à chaque variable de pile active au niveau de cette instruction ;
- à examiner chaque instruction détectée de rang i et à discriminer l'existence d'une cible de branchement respectivement d'un branchement, et dans le cas où l'instruction de rang i est une cible de branchement et que la pile d'exécution au niveau de cette instruction n'est pas vide,
 - pour toute instruction précédente, de rang i-1, constituée par un branchement, une levée d'exception ou un retour de programme, l'instruction détectée de rang i n'étant accessible que par un branchement,
 - à insérer un ensemble d'instructions de chargement load à partir de l'ensemble de nouveaux registres antérieurement à ladite instruction détectée de rang

i, avec redirection de tous les branchements vers l'instruction détectée de rang i vers la première instruction de chargement load insérée ; et

- pour toute instruction précédente, de rang i-1, continuant en séquence, l'instruction détectée de rang i étant accessible à la fois par l'intermédiaire d'un branchement et de l'instruction précédente de rang i-1,
- à insérer un ensemble d'instructions de sauvegarde store vers l'ensemble de nouveaux registres antérieurement à ladite instruction détectée de rang i et un ensemble d'instructions de chargement load à partir de cet ensemble de nouveaux registres, avec redirection de tous les branchements vers l'instruction détectée de rang i vers la première instruction de chargement load insérée, et, dans le cas où ladite instruction détectée de rang i est un branchement vers une instruction déterminée,
- pour toute instruction détectée de rang i constituée par un branchement inconditionnel,
- à insérer antérieurement à l'instruction détectée de rang i une pluralité d'instructions de sauvegarde store, à chaque nouveau registre étant associée une instruction de sauvegarde ; et
- pour toute instruction détectée de rang i constituée par un branchement conditionnel et pour un nombre $m > 0$ d'opérandes manipulés par cette instruction de branchement conditionnel,
- à insérer antérieurement à cette instruction détectée de rang i une instruction de permutation,

5 swap-x, au sommet de la pile d'exécution des m
opérandes de l'instruction détectée de rang i et des
n valeurs suivantes, cette opération de permutation
permettant de ramener au sommet de la pile
d'exécution les n valeurs à sauvegarder dans
l'ensemble des nouveaux registres, et

- à insérer antérieurement à l'instruction de rang i
un ensemble d'instructions de sauvegarde store vers
l'ensemble de nouveaux registres, et
- 10 •• à insérer postérieurement à l'instruction détectée
de rang i un ensemble d'instructions de chargement
load à partir de l'ensemble des nouveaux registres.

15 18. Procédé selon la revendication 16, caractérisé
en ce que l'étape consistant à effectuer une réallocation
des registres par détection des registres d'origine
employés avec des types différents consiste :

- à déterminer les intervalles de durée de vie de chaque
registre ;
- 20 - à déterminer le type de données principal de chaque
intervalle de durée de vie, le type de données
principal d'un intervalle de durée de vie j pour un
registre r étant défini par la borne supérieure des
types de données stockées dans ce registre r par les
instructions de sauvegarde store appartenant à
25 l'intervalle de durée de vie j ;
- à établir un graphe d'interférences entre les
intervalles de durée de vie, ce graphe d'interférences
consistant en un graphe non orienté dont chaque sommet
est constitué par un intervalle de durée de vie et dont
30 les arcs entre deux sommets j_1 et j_2 existent si un

sommet contient une instruction de sauvegarde adressée au registre de l'autre sommet ou réciproquement ;

- à traduire l'unicité d'un type de donnée alloué à chaque registre dans le graphe d'interférences en ajoutant des arcs entre toute paire de sommets du graphe d'interférences tant que deux sommets d'une paire de sommets n'ont pas le même type de donnée principal associé ;

- à effectuer une instanciación du graphe d'interférences, par attribution à chaque intervalle de durée de vie d'un numéro de registre de telle manière qu'à deux intervalles de vie adjacents dans le graphe d'interférences soient attribués des numéros de registres différents.

19. Système embarqué reprogrammable par téléchargement de fragments de programmes, comportant au moins un microprocesseur, une mémoire vive, un module d'entrées/sorties, une mémoire non volatile reprogrammable électriquement et une mémoire permanente dans laquelle sont implantés un programme principal et une machine virtuelle permettant l'exécution du programme principal et d'au moins un fragment de programme par l'intermédiaire dudit microprocesseur, caractérisé en ce que ledit système embarqué comporte au moins un module de programme de gestion et de vérification d'un fragment de programme téléchargé, ledit module de programme de gestion et de vérification étant implanté en mémoire permanente.

20. Système embarqué reprogrammable par téléchargement de fragments de programmes, comportant au moins un microprocesseur, une mémoire vive, un module d'entrées/sorties, une mémoire non volatile reprogrammable

électriquement et une mémoire permanente dans laquelle sont implantés un programme principal et une machine virtuelle permettant l'exécution du programme principal et d'au moins un fragment de programme par l'intermédiaire dudit microprocesseur, caractérisé en ce que ledit système embarqué comporte au moins un module de programme de gestion et de vérification d'un fragment de programme téléchargé suivant le protocole de gestion d'un fragment de programme téléchargé selon l'une des revendications 1 à 3, ledit module de programme de gestion et de vérification étant implanté en mémoire permanente.

21. Système embarqué selon la revendication 20, caractérisé en ce que celui-ci comporte au moins un module de sous-programme de vérification d'un fragment de programme téléchargé, suivant le processus de vérification selon l'une des revendications 4 à 14.

22. Système de transformation d'un code objet d'un fragment de programme, dans lequel les opérandes de chaque instruction appartiennent aux types de données manipulées par cette instruction, la pile d'exécution ne présente pas de phénomène de débordement, pour chaque instruction de branchement le type de variables de pile au niveau de ce branchement est le même qu'au niveau des cibles de ce branchement et un opérande de type déterminé écrit dans un registre par une instruction de ce code objet est relu depuis ce même registre par une autre instruction de ce code objet avec le même type de donnée déterminé, en un code objet normalisé pour ce même fragment de programme dans lequel les opérandes de chaque instruction appartiennent aux types de données manipulées par cette instruction, la pile d'exécution ne présente pas de

phénomène de débordement, la pile d'exécution est vide à chaque instruction de branchement et à chaque instruction de cible de branchement, un seul et même type de donnée étant alloué à un même registre dans tout ledit code objet normalisé, caractérisé en ce que ledit système de transformation comporte au moins, implanté en mémoire de travail d'un ordinateur de développement ou d'une station de travail, un module de programme de transformation de ce code objet en un code objet normalisé suivant le procédé selon l'une des revendications 15 à 18, ce qui permet d'engendrer un code objet normalisé pour ledit fragment de programme satisfaisant aux critères de vérification de ce fragment de programme téléchargé.

23. Un produit programme d'ordinateur chargeable directement dans la mémoire interne d'un système embarqué reprogrammable, tel qu'une carte à microprocesseur munie d'une mémoire réinscriptible, ce système embarqué permettant le téléchargement d'un fragment de programme constitué par un code objet, suite d'instructions, exécutable par le microprocesseur du système embarqué par l'intermédiaire d'une machine virtuelle munie d'une pile d'exécution et de registres ou variables locales manipulées par ces instructions et permettant d'interpréter ce code objet, ce produit programme d'ordinateur comprenant des portions de code objet pour l'exécution du protocole de gestion d'un fragment de programme téléchargé sur ce système embarqué selon l'une des revendications 1 à 3, lorsque ce système embarqué est interconnecté à un terminal et que ce programme est exécuté par le microprocesseur de ce système embarqué par l'intermédiaire de ladite machine virtuelle.

24. Un produit programme d'ordinateur chargeable directement dans la mémoire interne d'un système embarqué reprogrammable, tel qu'une carte à microprocesseur munie d'une mémoire réinscriptible, ce système embarqué
5 permettant le téléchargement d'un fragment de programme constitué par un code objet, suite d'instructions, exécutable par le microprocesseur du système embarqué par l'intermédiaire d'une machine virtuelle munie d'une pile d'exécution et de registres d'opérandes manipulés par ces
10 instructions et permettant d'interpréter ce code objet, ce produit programme d'ordinateur comprenant des portions de code objet pour l'exécution des étapes de vérification d'un fragment de programme téléchargé sur ce système embarqué selon l'une des revendications 4 à 14 lorsque ce
15 système embarqué est interconnecté à un terminal et que ce programme est exécuté par le microprocesseur de ce système embarqué par l'intermédiaire de ladite machine virtuelle.

25. Un produit programme d'ordinateur comprenant des portions de code objet pour l'exécution des étapes du
20 procédé de transformation d'un code objet d'un fragment de programme téléchargé en un code objet normalisé pour ce même fragment de programme selon l'une des revendications 15 à 18.

26. Un produit programme d'ordinateur enregistré
25 sur un support utilisable dans un système embarqué reprogrammable, tel qu'une carte à microprocesseur munie d'une mémoire réinscriptible, ce système embarqué permettant le téléchargement d'un fragment de programme constitué par un code objet, suite d'instructions,
30 exécutable par le microprocesseur du système embarqué par l'intermédiaire d'une machine virtuelle munie d'une pile

d'exécution et de registres ou variables locales manipulées par ces instructions et permettant d'interpréter ce code objet, ce produit programme d'ordinateur comprenant au moins :

- 5 - des moyens de programmes lisibles par le microprocesseur de ce système embarqué par l'intermédiaire de ladite machine virtuelle, pour commander l'exécution d'une procédure de gestion du téléchargement d'un fragment de programme téléchargé ;
- 10 - des moyens de programmes lisibles par le microprocesseur de ce système embarqué par l'intermédiaire de ladite machine virtuelle, pour commander l'exécution d'une procédure de vérification instruction par instruction du code objet constitutif
- 15 dudit fragment de programme ;
- des moyens de programmes lisibles par le microprocesseur de ce système embarqué par l'intermédiaire de ladite machine virtuelle pour commander l'exécution d'un fragment de programme
- 20 téléchargé suite à ou en l'absence d'une transformation du code objet de ce fragment de programme en code objet normalisé pour ce même fragment de programme.

27. Un produit programme d'ordinateur selon la revendication 26, comprenant en outre des moyens de programmes lisibles par le microprocesseur de ce système embarqué par l'intermédiaire de ladite machine virtuelle pour commander l'inhibition de l'exécution, sur ledit système embarqué, dudit fragment de programme dans le cas d'une procédure de vérification non réussie de ce fragment

30 de programme.

THIS PAGE BLANK (USPTO)

1/14

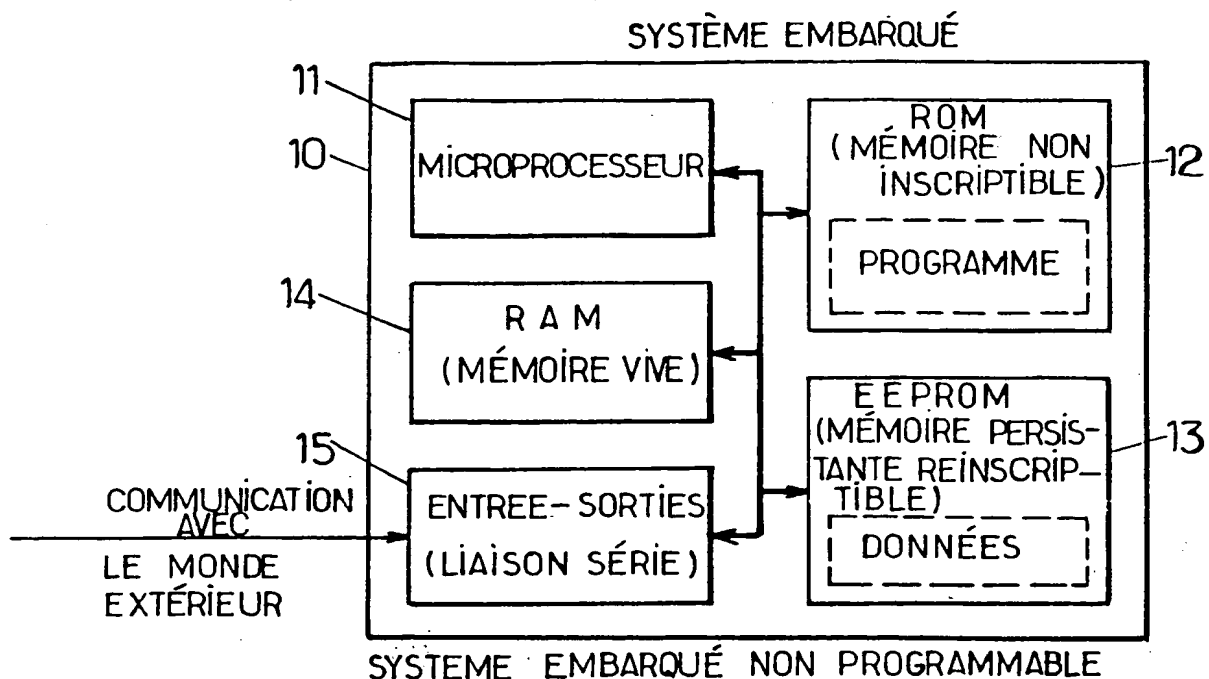


FIG.1a. (ART ANTÉRIEUR)

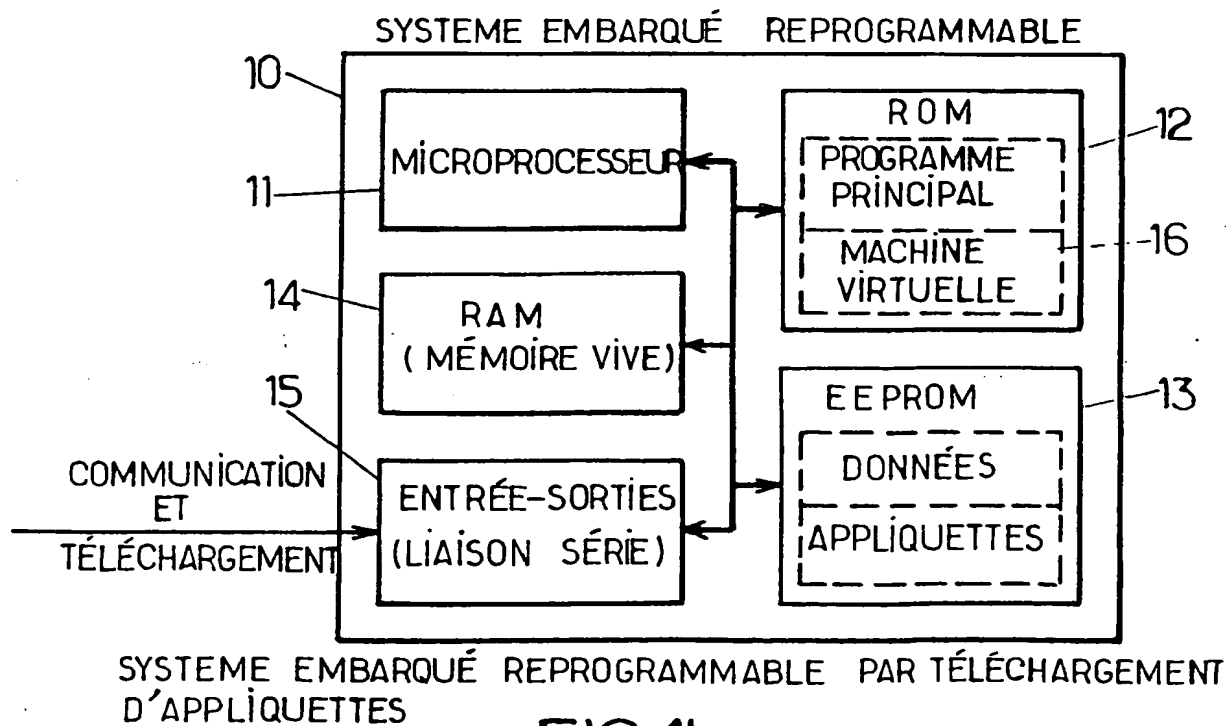


FIG.1b. (ART ANTERIEUR)

THIS PAGE BLANK (USPTO)

2/14

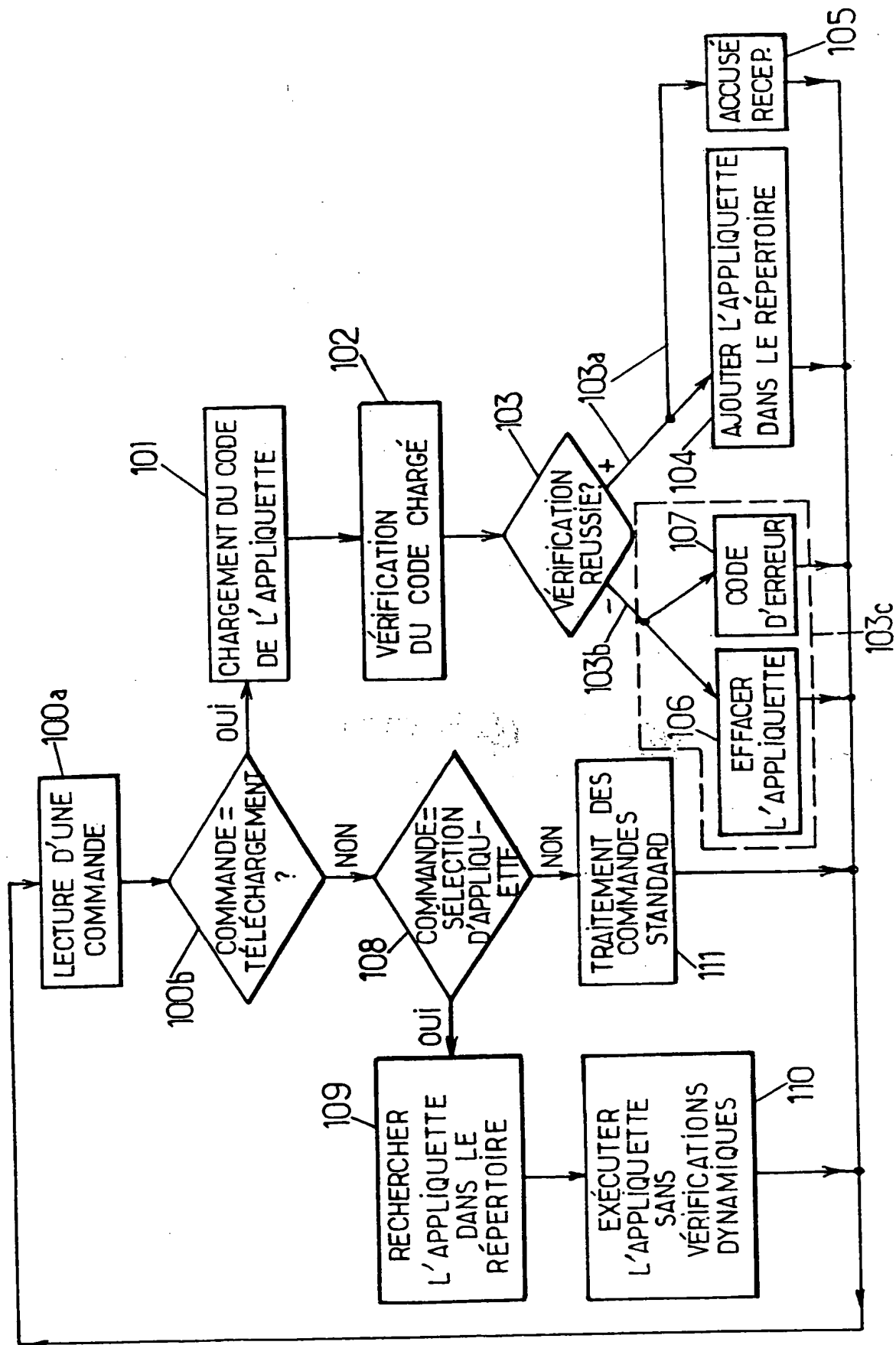


FIG. 2. PROTOCOLE DE GESTION

THIS PAGE BLANK (USPTO)

3/14

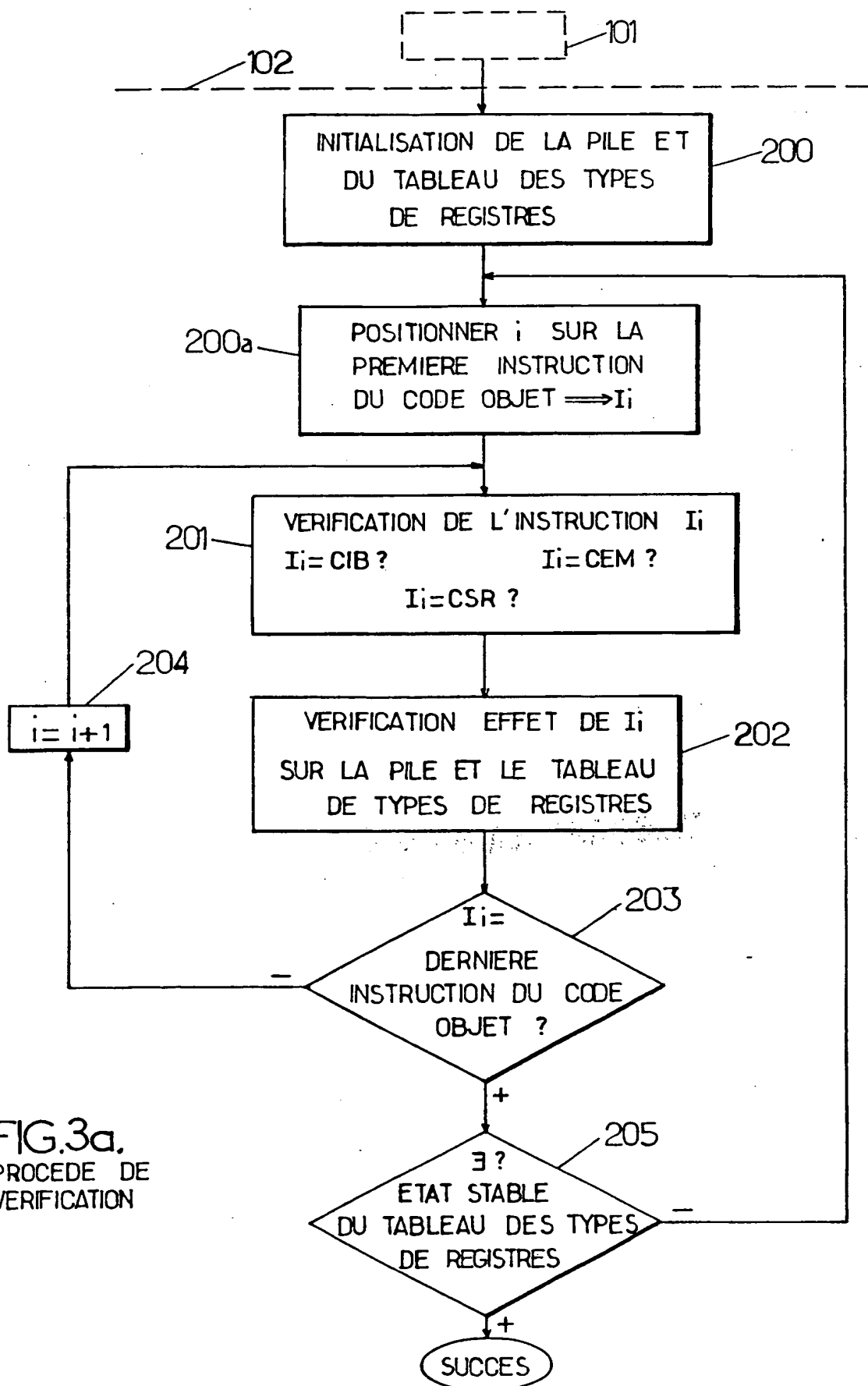


FIG. 3a.
PROCEDE DE
VERIFICATION

THIS PAGE BLANK (USPTO)

4/14

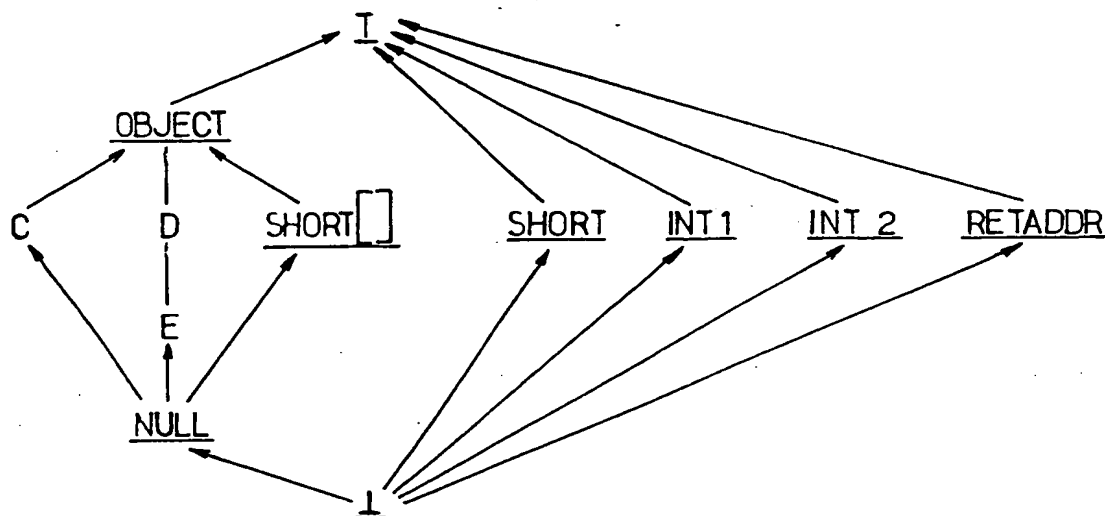


FIG.3b. TYPES DE DONNEES ET RELATION DE SOUS-TYPAGE

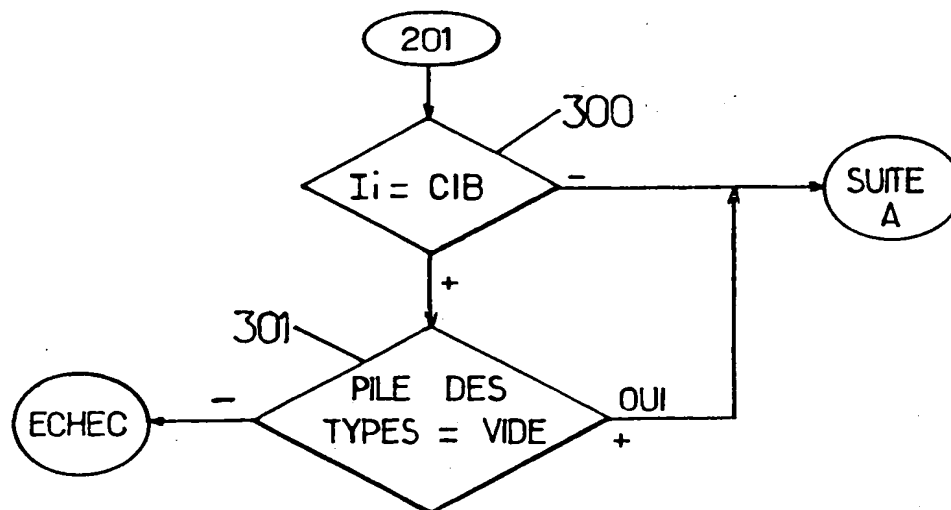
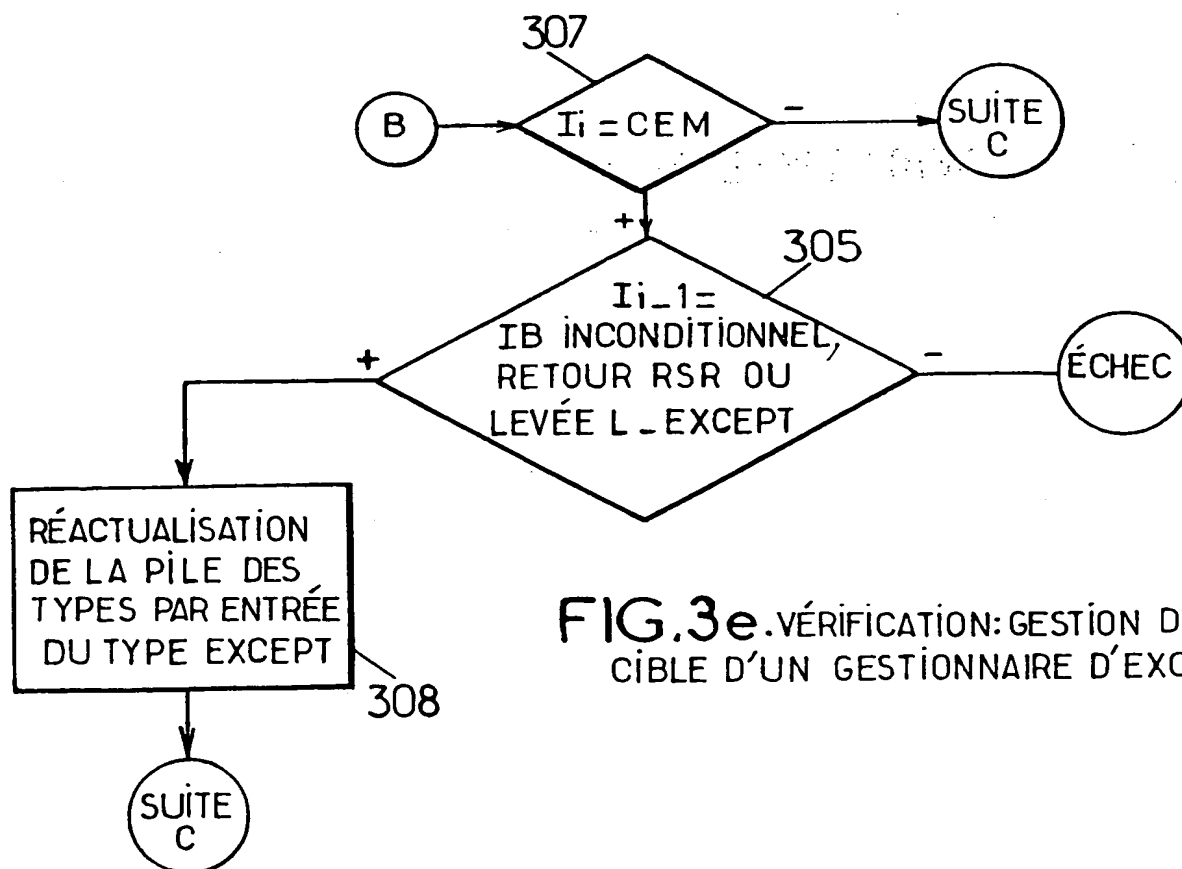
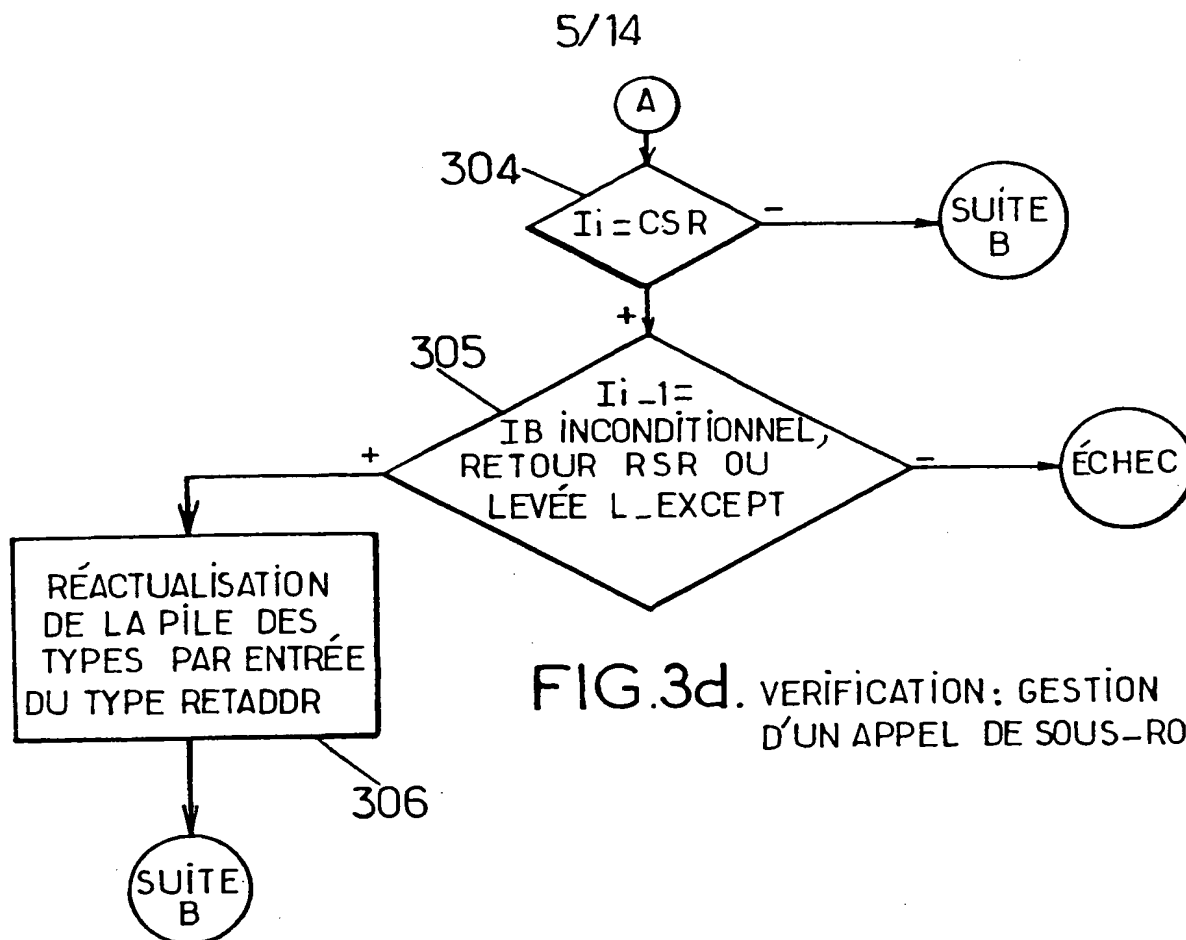


FIG.3c. VERIFICATION: GESTION D'UNE INSTRUCTION DE BRANCHEMENT

THIS PAGE BLANK (USPTO)



THIS PAGE BLANK (USPTO)

6/14

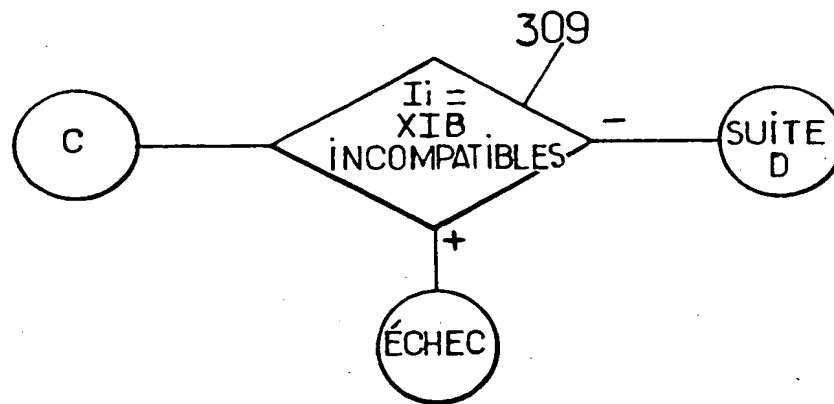


FIG. 3f. VÉRIFICATION: GESTION DE LA CIBLE DE BRANCHEMENTS INCOMPATIBLES

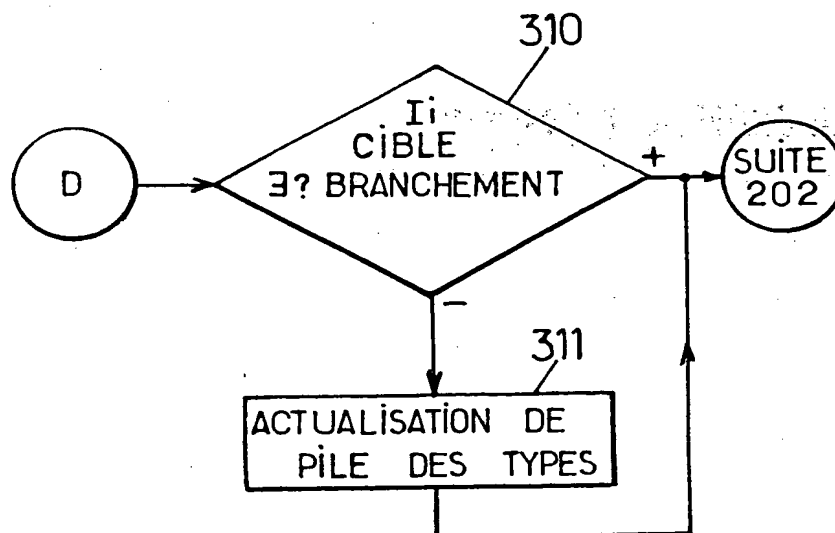


FIG. 3g. VÉRIFICATION: GESTION DE L'ABSENCE DE CIBLE DE BRANCHEMENT

THIS PAGE BLANK (USPTO)

7/14

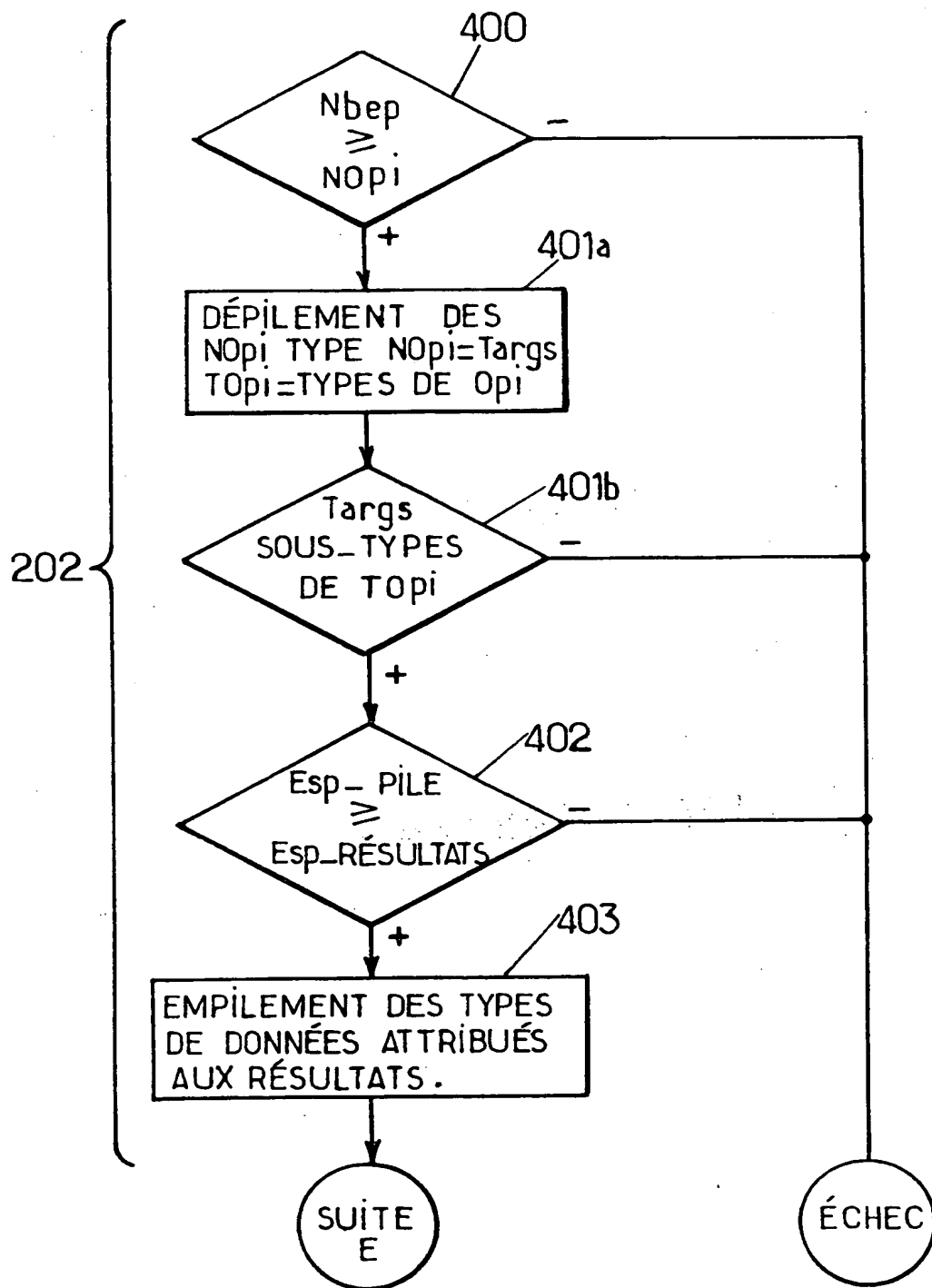


FIG.3h. VÉRIFICATION: EFFET DE L'INSTRUCTION COURANTE
SUR LA PILE DES TYPES

THIS PAGE BLANK (USPTO)

8/14

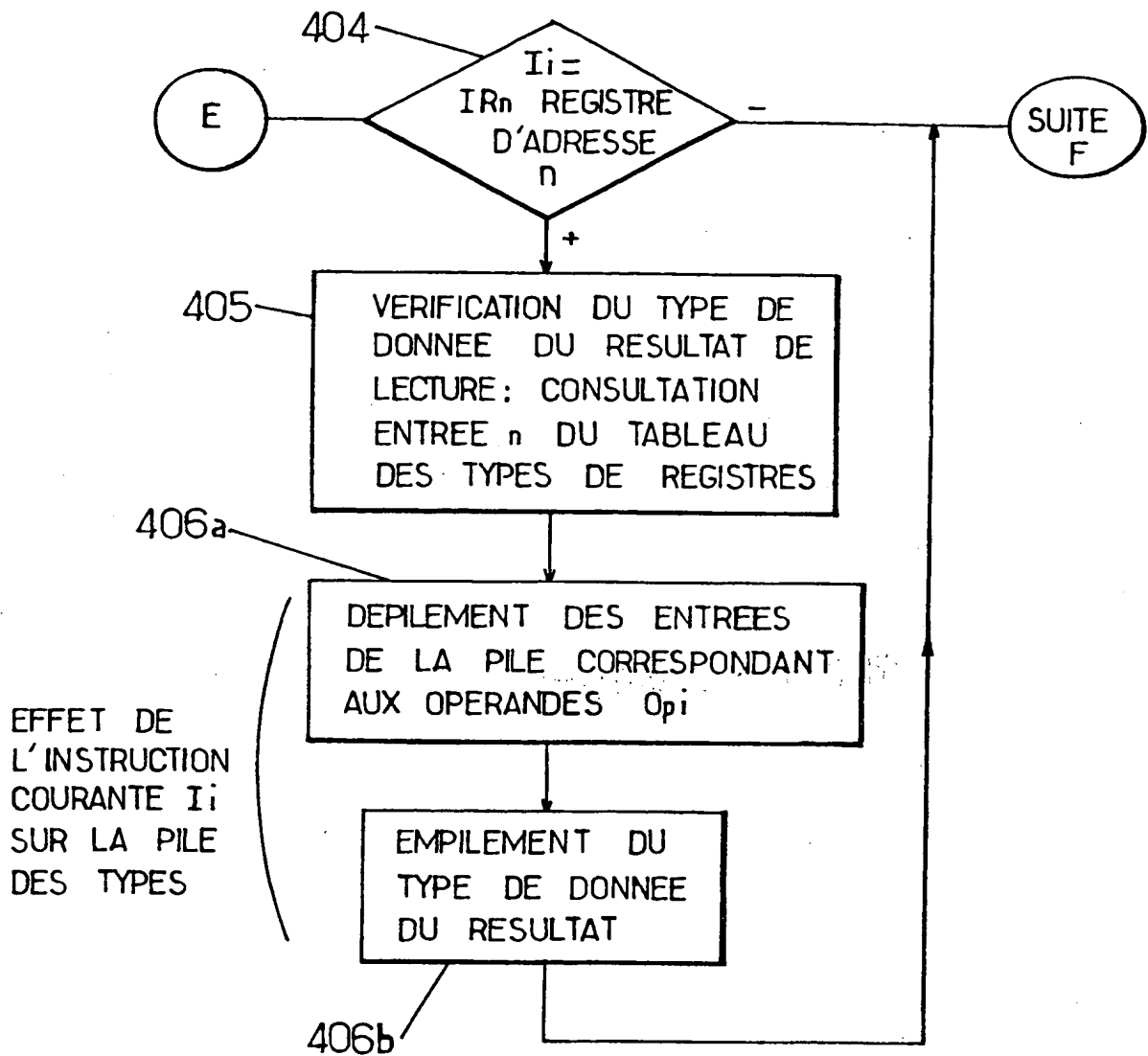


FIG.3i. VERIFICATION: GESTION D'UNE INSTRUCTION DE LECTURE D'UN REGISTRE

THIS PAGE BLANK (USPTO)

9/14

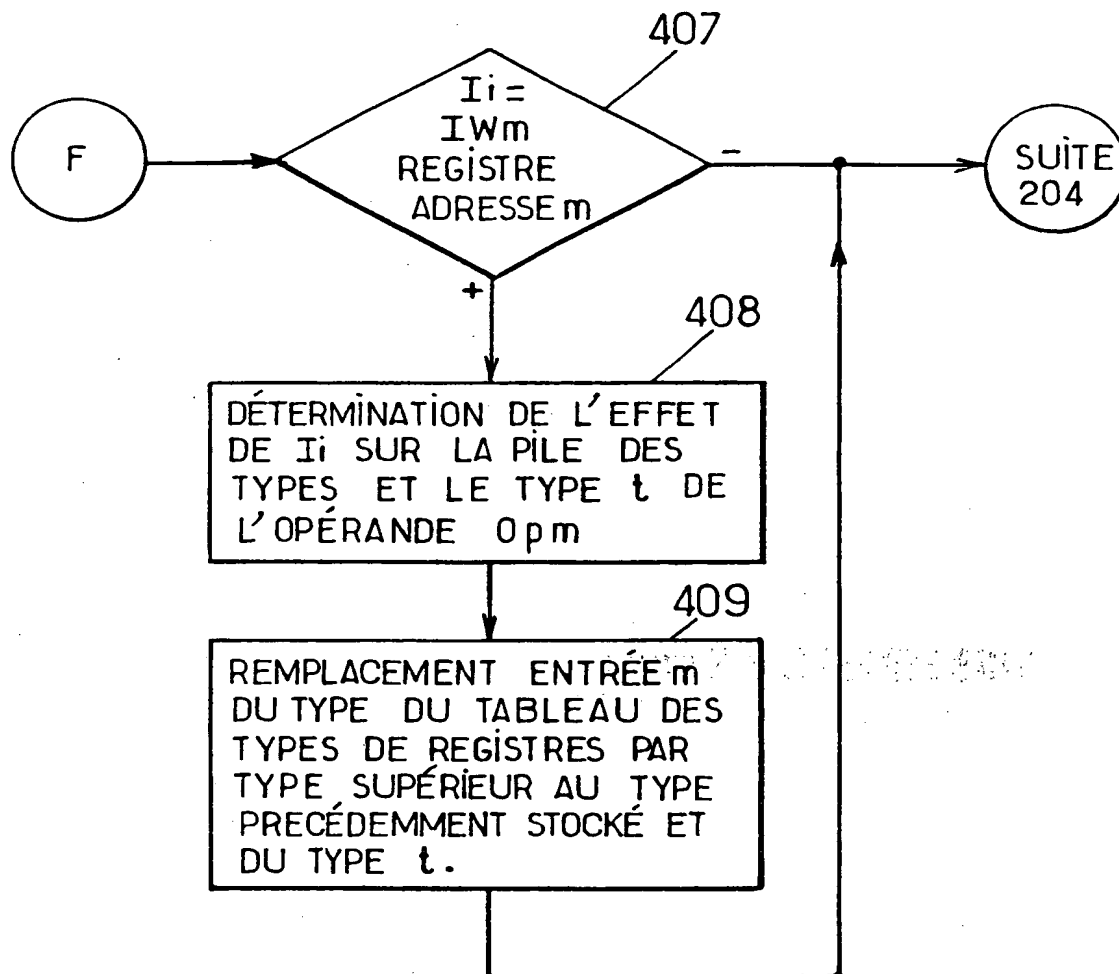


FIG. 3j. VÉRIFICATION: GESTION D'UNE INSTRUCTION D'ÉCRITURE D'UN REGISTRE

THIS PAGE BLANK (USPTO)

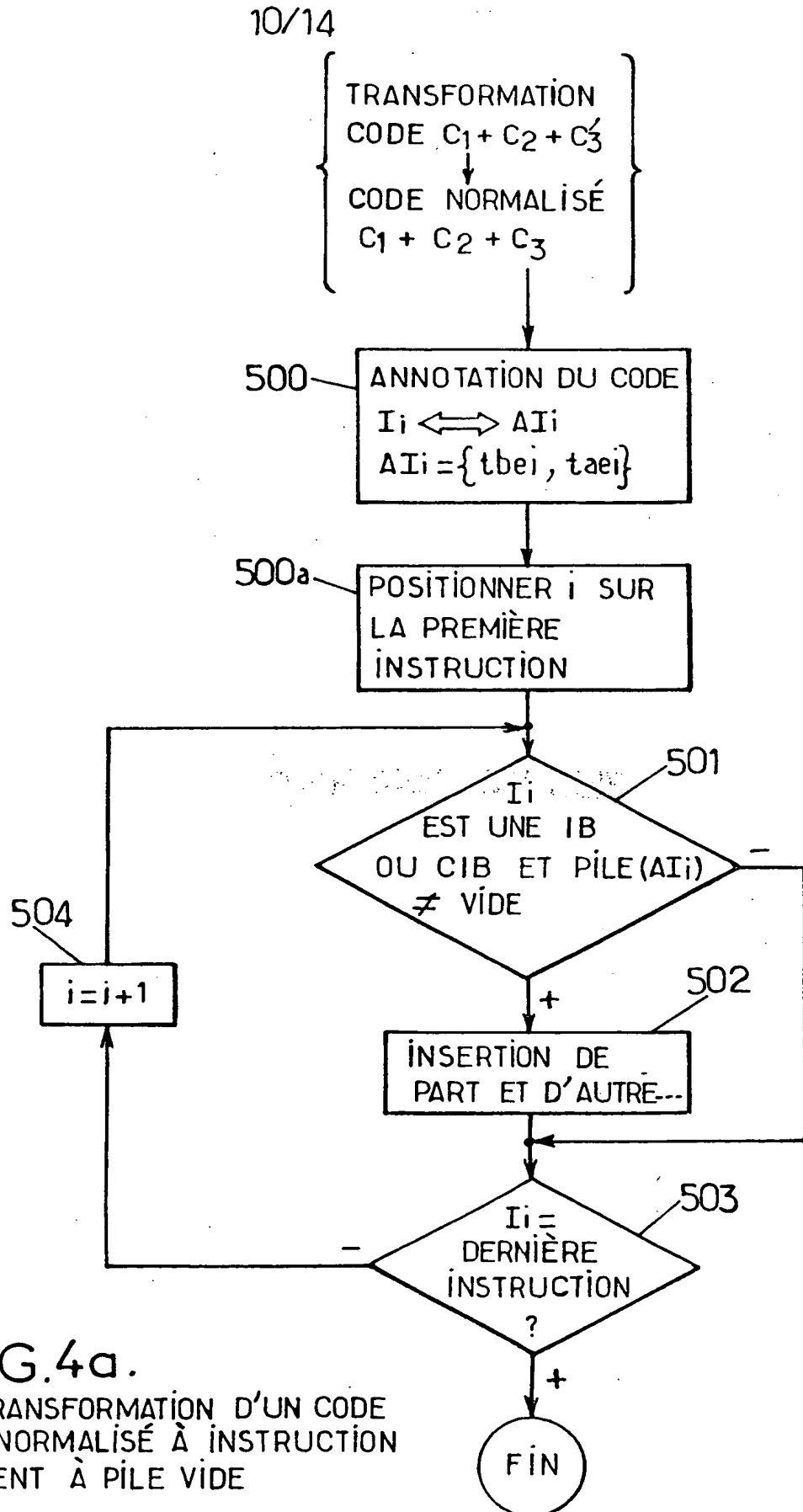


FIG.4a.

PROCÉDÉ DE TRANSFORMATION D'UN CODE
EN UN CODE NORMALISÉ À INSTRUCTION
DE BRANCHEMENT À PILE VIDE

1 3

THIS PAGE BLANK (USPTO)

11/14

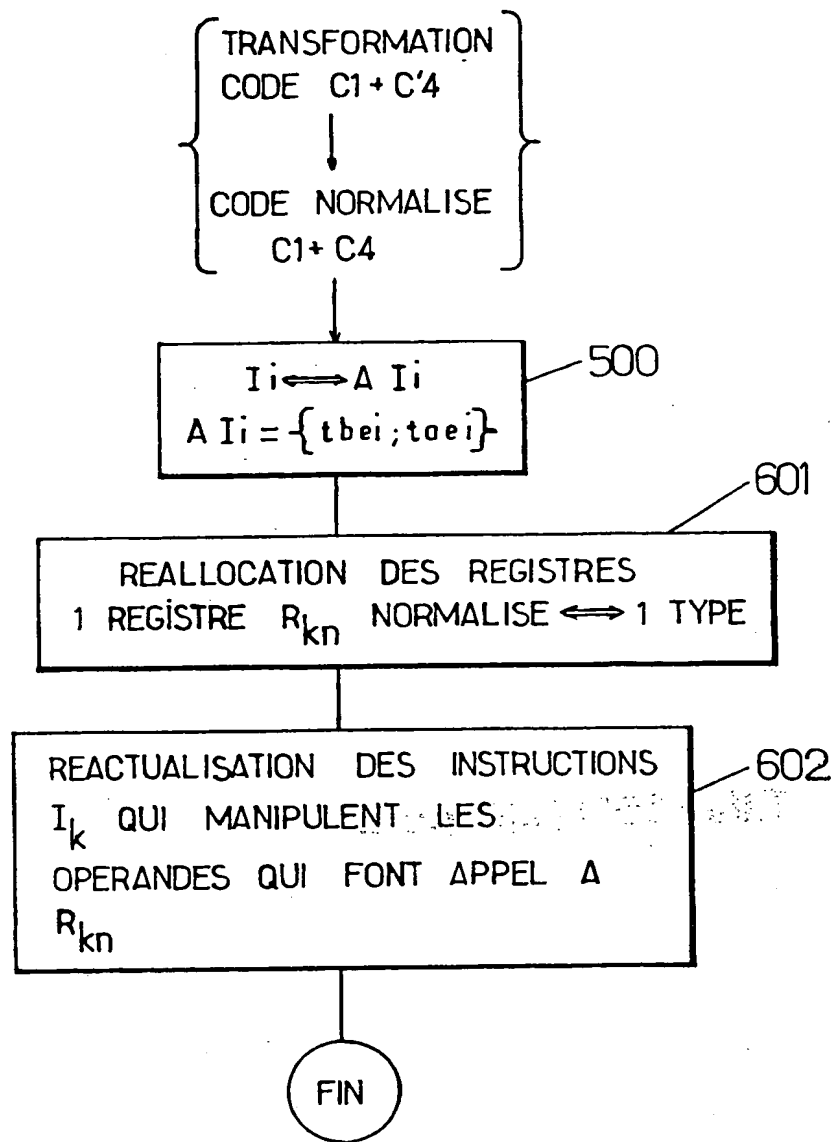
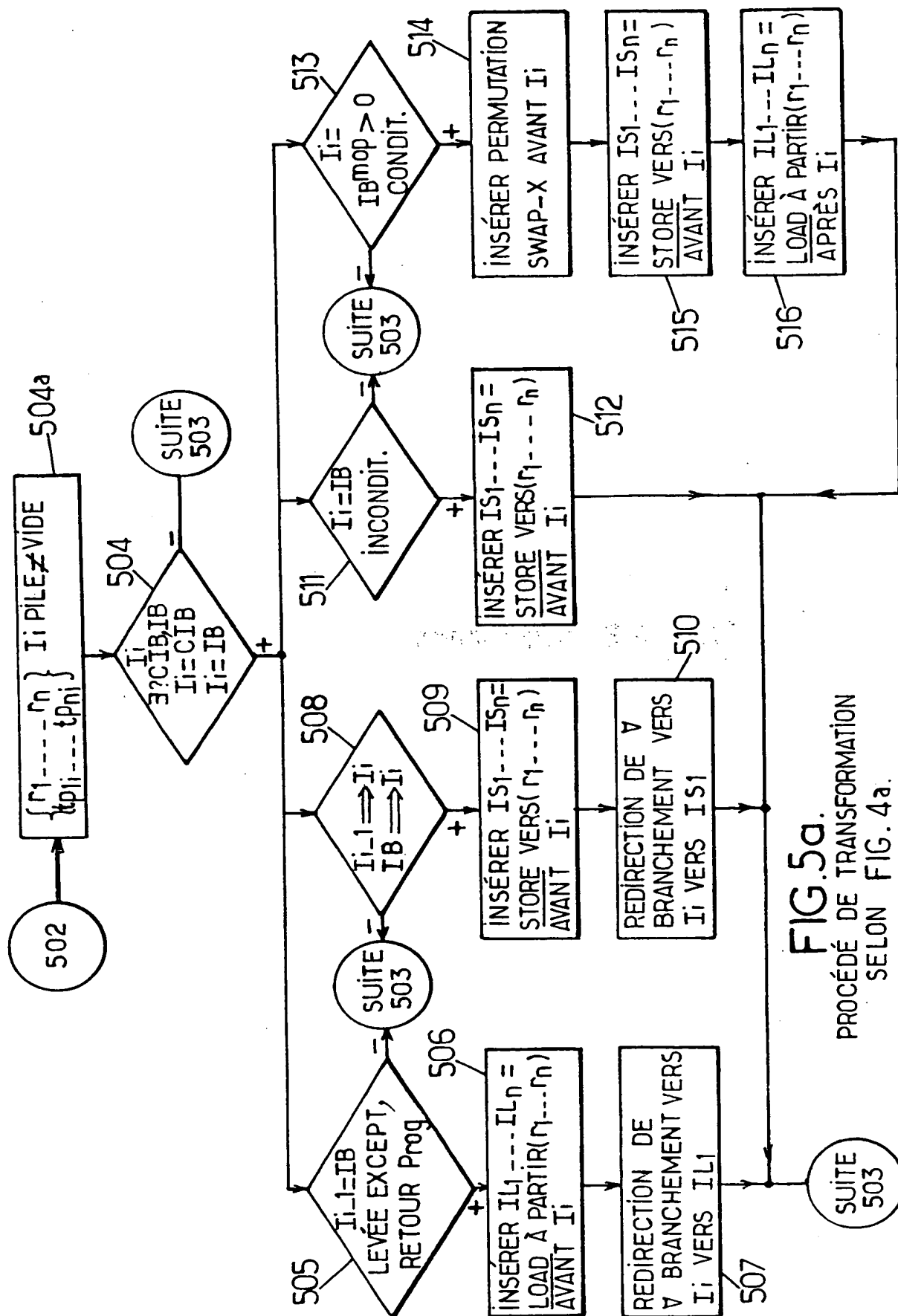


FIG.4b. PROCEDE DE TRANSFORMATION D'UN CODE
EN UN CODE NORMALISE FAISANT APPEL
A DES REGISTRES TYPE

THIS PAGE BLANK (USPTO)

12/14



THIS PAGE BLANK (USPTO)

13/14

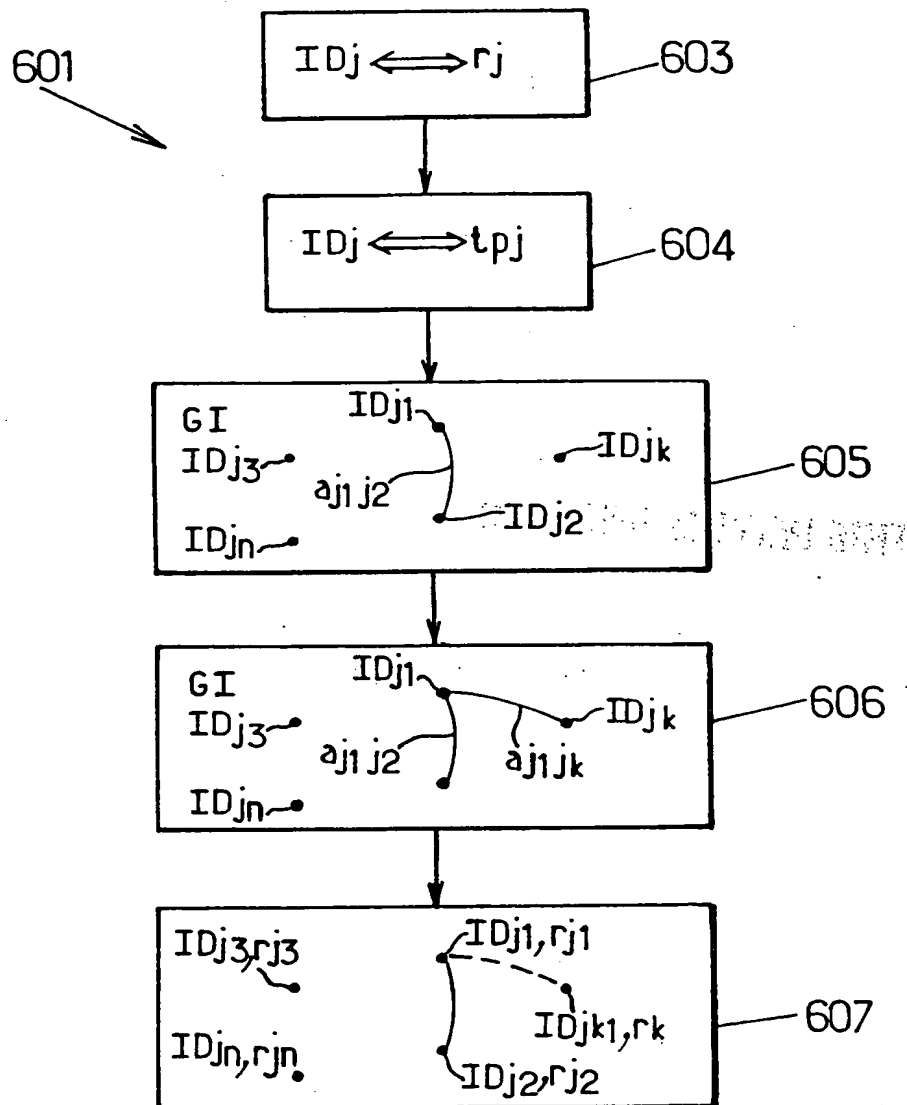


FIG.5b.
PROCÉDÉ DE TRANSFORMATION
SELON LA FIG.4b.

THIS PAGE BLANK (USPTO)

14/14

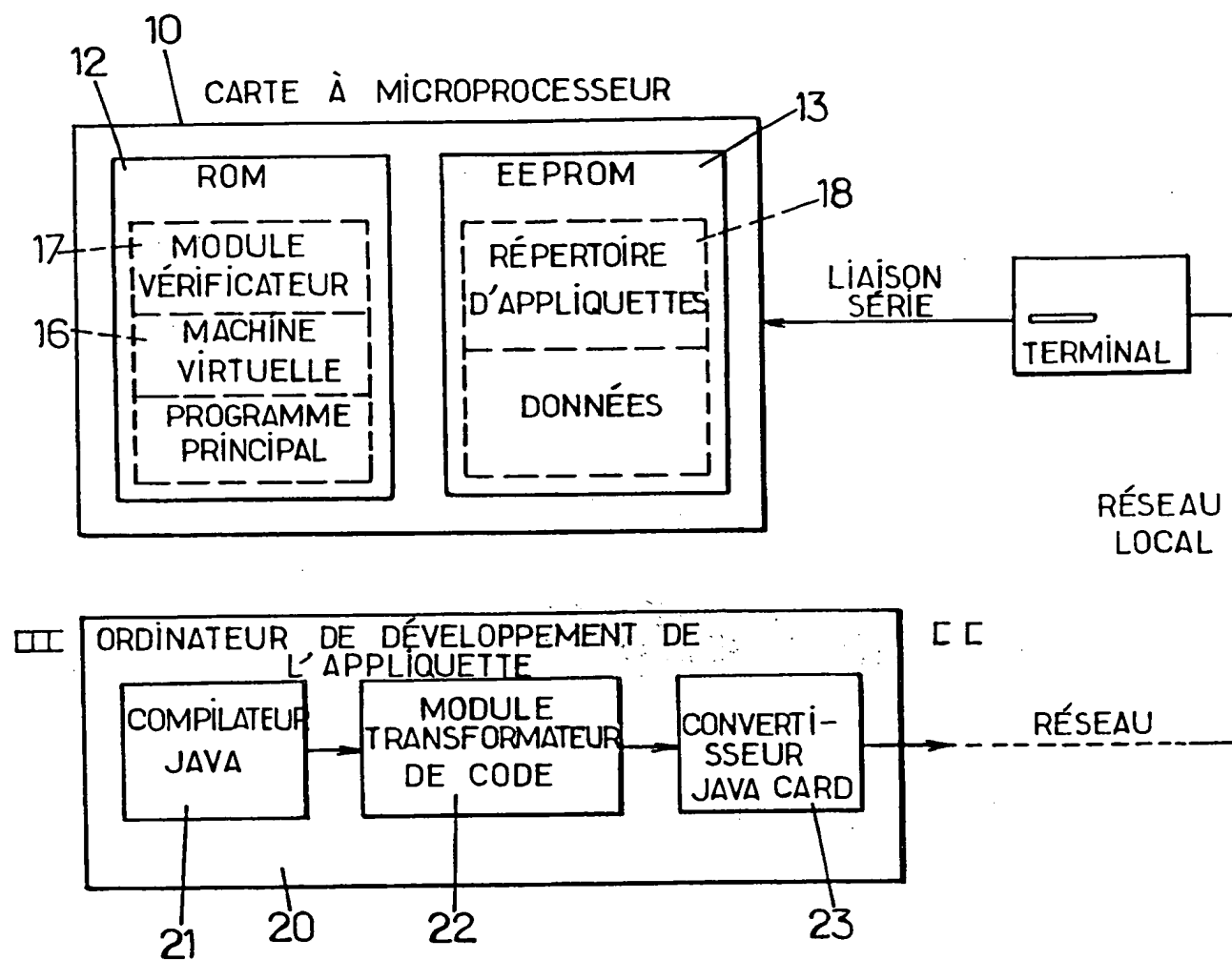


FIG. 6.

THIS PAGE BLANK (USPTO)

(12) DEMANDE INTERNATIONALE PUBLIÉE EN VERTU DU TRAITÉ DE COOPÉRATION
EN MATIÈRE DE BREVETS (PCT)

(19) Organisation Mondiale de la Propriété
Intellectuelle
Bureau international



(43) Date de la publication internationale
1 mars 2001 (01.03.2001)

PCT

(10) Numéro de publication internationale
WO 01/14958 A3

(51) Classification internationale des brevets⁷ :
G06F 9/445, 9/45

(71) Déposant (pour tous les États désignés sauf US) :
TRUSTED LOGIC [FR/FR]; 5, rue du Bailliage,
F-78000 Versailles (FR).

(21) Numéro de la demande internationale :
PCT/FR00/02349

(72) Inventeur; et
(75) Inventeur/Déposant (pour US seulement) : LEROY,
Xavier [FR/FR]; 88 bis, avenue de Paris, F-78000 Ver-
sailles (FR).

(22) Date de dépôt international : 21 août 2000 (21.08.2000)

(25) Langue de dépôt : français

(74) Mandataires : FRECHEDE, Michel etc.; Cabinet
Plasseraud, 84, rue d'Amsterdam, F-75440 Paris Cedex 09
(FR).

(26) Langue de publication : français

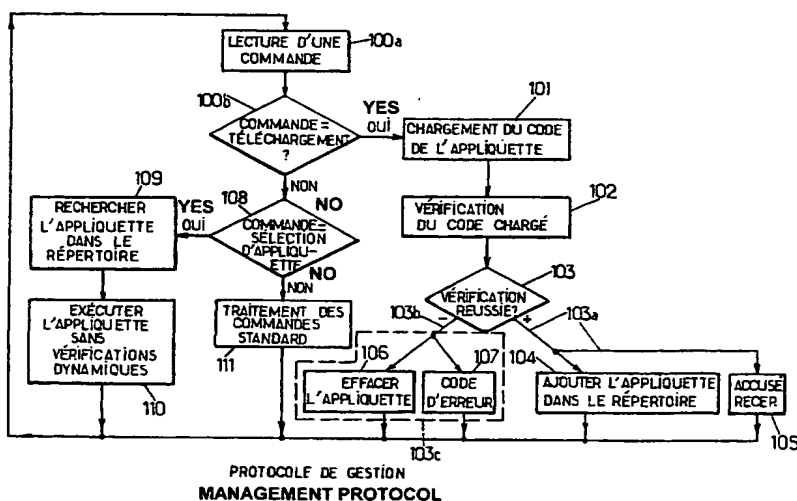
(30) Données relatives à la priorité :
99/10697 23 août 1999 (23.08.1999) FR

(81) États désignés (national) : AU, CA, CN, JP, US.

[Suite sur la page suivante]

(54) Title: MANAGEMENT PROTOCOL, METHOD FOR VERIFYING AND TRANSFORMING A DOWNLOADED PRO-
GRAMME FRAGMENT AND CORRESPONDING SYSTEMS

(54) Titre : PROTOCOLE DE GESTION, PROCEDE DE VERIFICATION ET DE TRANSFORMATION D'UN FRAGMENT
DE PROGRAMME TELECHARGE ET SYSTEMES CORRESPONDANTS



100a ... COMMAND READ
100b ... COMMAND=DOWNLOAD?
101 ... APPLET CODE DOWNLOAD
102 ... DOWNLOADED CODE VERIFIED
103 ... VERIFICATION SUCCESSFUL?
105 ... RECEIVE ACKNOWLEDGEMENT
104 ... ADD APPLET TO REPTURE
107 ... ERROR CODE
106 ... DELETE APPLET
111 ... STANDARD COMMAND
110 ... EXECUTE APPLET WITHOUT DYNAMIC CHECKS
109 ... SEARCH FOR APPLET IN REPECTOIRE
108 ... COMMAND=SELECTION OF APPLET

(57) Abstract: The invention relates to a management protocol and to a method for verifying a programme fragment, or applet, which has been downloaded onto a portable system. An applet downloading command (100a, 100b) is executed. Once a positive response has been received, the object code of the applet is read (101) and subjected (102) to a verification process, instruction by instruction. The verification process consists of a stage comprising the initialisation of the type stack and table of register types representing the state of the virtual machine of the portable system at the start of the execution of the applet code; and a verification, instruction by instruction, for each target current instruction, of the existence of a target branch instruction, a target exception handler call or a target sub-routine call, the effect of the instruction on the type stack and the table of register types being verified and updated. If the verification is successful (103a), the applet is registered (104) and an acknowledgement is sent (105) to the downloading drive. Otherwise, the applet

[Suite sur la page suivante]

WO 01/14958 A3



(84) États désignés (*régional*) : brevet européen (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE).

Publiée :

— avec rapport de recherche internationale

(88) Date de publication du rapport de recherche internationale:

13 décembre 2001

En ce qui concerne les codes à deux lettres et autres abréviations, se référer aux "Notes explicatives relatives aux codes et abréviations" figurant au début de chaque numéro ordinaire de la Gazette du PCT.

is destroyed (106). The invention is suitable for use for portable systems in a Java environment.

(57) **Abrégé :** L'invention concerne un protocole de gestion et un procédé de vérification d'un fragment de programme, ou applique, téléchargé sur un système embarqué. Une commande de téléchargement (100a, 100b) de l'applique est effectuée. Sur réponse positive, le code objet de l'applique est lu (101) et soumis (102) à une vérification instruction par instruction. La vérification consiste en une étape d'initialisation de la pile des types et du tableau des types de registres représentant l'état de la machine virtuelle du système embarqué au début de l'exécution du code de l'applique et en une vérification, instruction par instruction pour chaque instruction courante cible, de l'existence d'une cible d'instruction de branchement, d'appel d'un gestionnaire d'exceptions ou d'un appel de sous-routine. et par une vérification et une actualisation de l'effet de cette instruction sur la pile des types et le tableau des types de registres. Sur vérification réussie (103a), l'applique est enregistrée (104) et un accusé de réception est envoyé (105) au lecteur de téléchargement. L'applique est détruite (106) sinon. Application aux systèmes embarqués en environnement Java.

INTERNATIONAL SEARCH REPORT

Int. Application No
PCT/Fr 00/02349

A. CLASSIFICATION OF SUBJECT MATTER
IPC 7 G06F9/445 G06F9/45

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

IPC 7 G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)

EPO-Internal, PAJ, INSPEC, COMPENDEX

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	US 5 740 441 A (GOSLING JAMES A ET AL) 14 April 1998 (1998-04-14)	1-5,7,8, 10-13, 19,20, 23,24
A	column 1, line 1 -column 15, line 11	15,16, 21,25
X	column 17, line 7 - line 10 column 18, line 21 - line 23 column 18, line 32 - line 34	26,27
Y	GOSLING J ET AL: "THE JAVATM LANGUAGE SPECIFICATION", JAVA LANGUAGE SPECIFICATION,XX,XX, PAGE(S) 215-236 XP002042923	1-5,7,8, 10-13, 19,20, 23,24
A	page 218, line 10 -page 220, line 35 --- -/--	6,9

☒ Further documents are listed in the continuation of box C.

☒ Patent family members are listed in annex.

* Special categories of cited documents :

- *A* document defining the general state of the art which is not considered to be of particular relevance
- *E* earlier document but published on or after the international filing date
- *L* document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
- *O* document referring to an oral disclosure, use, exhibition or other means
- *P* document published prior to the international filing date but later than the priority date claimed

- *T* later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
- *X* document of particular relevance: the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
- *Y* document of particular relevance: the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.
- *Z* document member of the same patent family

Date of the actual completion of the international search

11 June 2001

Date of mailing of the international search report

19/06/2001

Name and mailing address of the ISA

European Patent Office, P.B. 5818 Patentlaan 2
NL - 2280 HV Rijswijk
Tel. (+31-70) 340-2040, Tx. 31 651 epo nl,
Fax: (+31-70) 340-3016

Authorized officer

Kingma, Y

INTERNATIONAL SEARCH REPORT

Application No
PCT/FR 00/02349

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	GONG L ET AL: "Going beyond the sandbox: an overview of the new security architecture in the JavaDevelopment Kit 1.2" , PROCEEDINGS OF THE USENIX SYMPOSIUM ON INTERNET TECHNOLOGIES AND SYSTEMS XP002100907 page 107, right-hand column, line 1 -page 108, left-hand column, line 3 -----	3
A	GUTHERY: "JAVA CARD: Internet Computing on a Smart Card" IEEE INTERNET COMPUTING,US,IEEE SERVICE CENTER, PISCATAWAY, NJ, 1 February 1997 (1997-02-01), pages 57-59, XP002077647 ISSN: 1089-7801 page 58, right-hand column, line 19 -page 59, middle column, line 24 -----	1,4,19, 26

INTERNATIONAL SEARCH REPORT

Inventor's name and patent family members

International Application No

PCT/FR 00/02349

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 5740441 A	14-04-1998	US 5748964 A	05-05-1998
		US 5999731 A	07-12-1999
		EP 0718764 A	26-06-1996
		JP 8234994 A	13-09-1996

THIS PAGE BLANK (USPTO)

RAPPORT DE RECHERCHE INTERNATIONALE

Deposition internationale No

PCT/FK 00/02349

A. CLASSEMENT DE L'OBJET DE LA DEMANDE
CIB 7 G06F9/445 G06F9/45

Selon la classification internationale des brevets (CIB) ou a la fois selon la classification nationale et la CIB

B. DOMAINES SUR LESQUELS LA RECHERCHE A PORTE

Documentation minimale consultee (systeme de classification suivi des symboles de classement)

CIB 7 G06F

Documentation consultee autre que la documentation minimale dans la mesure ou ces documents relevent des domaines sur lesquels a porte la recherche

Base de donnees electronique consultee au cours de la recherche internationale (nom de la base de donnees, et si realisable, termes de recherche utilises)

EPO-Internal, PAJ, INSPEC, COMPENDEX

C. DOCUMENTS CONSIDERES COMME PERTINENTS

Categorie *	Identification des documents cites, avec, le cas echeant, l'indication des passages pertinents	no. des revendications visees
Y	US 5 740 441 A (GOSLING JAMES A ET AL) 14 avril 1998 (1998-04-14)	1-5,7,8, 10-13, 19,20, 23,24
A	colonne 1, ligne 1 -colonne 15, ligne 11	15,16, 21,25
X	colonne 17, ligne 7 - ligne 10 colonne 18, ligne 21 - ligne 23 colonne 18, ligne 32 - ligne 34	26,27
Y	GOSLING J ET AL: "THE JAVATM LANGUAGE SPECIFICATION", JAVA LANGUAGE SPECIFICATION,XX,XX, PAGE(S) 215-236 XP002042923	1-5,7,8, 10-13, 19,20, 23,24
A	page 218, ligne 10 -page 220, ligne 35 --- -/--	6,9

☒ Voir la suite du cadre C pour la fin de la liste des documents

☒ Les documents de familles de brevets sont indiques en annexe

* Categories speciales de documents cites:

- *A* document definissant l'etat general de la technique, non considere comme particulierement pertinent
- *E* document anterieur, mais publie a la date de depot international ou apres cette date
- *L* document pouvant jeter un doute sur une revendication de priorite ou cite pour determiner la date de publication d'une autre citation ou pour une raison speciale (telle qu'indiquee)
- *O* document se referant a une divulgation orale, a un usage, a une exposition ou tous autres moyens
- *P* document publie avant la date de depot international, mais posterieurement a la date de priorite revendiquee

- *T* document ulterieur publie apres la date de depot international ou la date de priorite et n'appartenant pas a l'etat de la technique pertinent, mais cite pour comprendre le principe ou la theorie constituant la base de l'invention
- *X* document particulierement pertinent; l'invention revendiquee ne peut etre consideree comme nouvelle ou comme impliquant une activite inventive par rapport au document considere isolement
- *Y* document particulierement pertinent; l'invention revendiquee ne peut etre consideree comme impliquant une activite inventive lorsque le document est associe a un ou plusieurs autres documents de meme nature, cette combinaison etant evidente pour une personne du metier
- *Z* document qui fait partie de la meme famille de brevets

Date a laquelle la recherche internationale a ete effectivement achevee

11 juin 2001

Date d'expedition du present rapport de recherche internationale

19/06/2001

Nom et adresse postale de l'administration chargee de la recherche internationale
Office Européen des Brevets, P.B. 5818 Patentlaan 2
NL - 2280 HV Rijswijk
Tel. (+31-70) 340-2040, Tx. 31 651 epo nl.
Fax: (+31-70) 340-3016

Fonctionnaire autorise

Kingma, Y

RAPPORT DE RECHERCHE INTERNATIONALE

de Internationale No
PCT/FR 00/02349

C.(suite) DOCUMENTS CONSIDERES COMME PERTINENTS		
Catégorie	Identification des documents cités, avec, le cas échéant, l'indication des passages pertinents	no. des revendications visées
A	GONG L ET AL: "Going beyond the sandbox: an overview of the new security architecture in the Java Development Kit 1.2" , PROCEEDINGS OF THE USENIX SYMPOSIUM ON INTERNET TECHNOLOGIES AND SYSTEMS XP002100907 page 107, colonne de droite, ligne 1 -page 108, colonne de gauche, ligne 3 -----	3
A	GUTHERY: "JAVA CARD: Internet Computing on a Smart Card" IEEE INTERNET COMPUTING,US,IEEE SERVICE CENTER, PISCATAWAY, NJ, 1 février 1997 (1997-02-01), pages 57-59, XP002077647 ISSN: 1089-7801 page 58, colonne de droite, ligne 19 -page 59, colonne du milieu, ligne 24 -----	1,4,19, 26

RAPPORT DE RECHERCHE INTERNATIONALE

Renseignements relatifs aux familles de brevets

Demande internationale No

PCT/FK 00/02349

Document brevet cité au rapport de recherche	Date de publication	Membre(s) de la famille de brevet(s)	Date de publication
US 5740441 A	14-04-1998	US 5748964 A	05-05-1998
		US 5999731 A	07-12-1999
		EP 0718764 A	26-06-1996
		JP 8234994 A	13-09-1996

THIS PAGE BLANK (USPTO)